
# Opis zadania

Gliwice, 6-7 kwietnia 2010

# Spis treści

# Rozdział 1

# Wprowadzenie

Czy pamiętacie żukoskoczka jeżdżącego nieporadnie swoim pojazdem po powierzchniach nieznanych planet? Nieraz biedak roztrzaskał się o skały, czasem nie udało mu się dotrzeć do celu... No cóż, sterowanie pojazdem nie jest łatwe. Jednak mimo licznych niepowodzeń, żukoskoczki uparcie dążą do celu....

Realizacja misji podboju wszechświata, która przyświecała im w czasie penetracji odległych planet, posunęła się daleko naprzód. Żukoskoczki utworzyły już rozległe kolonie na kilku planetach. Kolonie takie składają się z niedużych miast, które łączą długie drogi. Drogi są tak długie, że żukoskoczki nie są w stanie poruszać się nimi o własnych siłach (a przynajmniej — nie mają ku temu dość motywacji). Jednak potrzeba przemieszczania się jest tak duża, że są w stanie słono płacić za przewóz. Oczywiście jako leniwe stworzonka nie lubią same kierować jakimikolwiek pojazdami (dlatego wszakże potrzebowały algorytmów do sterowania swoimi autkami w czasie eksploracji). Na szczęście — pojawiło się w okolicy wiele firm przewozowych. Twoja drużyna dowodzi jedną z nich.

Na każdej z planet kolonizowanych przez żukoskoczki Twoja firma posiada pewną ilość samochodów. I to samochodów nie byle jakich! Żukoskoczki są zainteresowane jedynie samochodami marki "Żuk". Legendy i podania ludowe (czy raczej: żukowe) głoszą, że były takowe produkowane kiedyś na planecie NBP (oficjalny skrót od Niegdyś Błękitna Planeta), w okolicy 51.240 °N, 22.570 °E wg. lokalnych współrzędnych (tamtejsi mieszkańcy nazywali to miejsce "Lublinem"). Muzea motoryzacji nie chciały jednak odsprzedawać swoich eksponatów wspomnianej marki, więc wobec rosnącego popytu, rozpoczęto produkcję replik. W miarę potrzeb zmodyfikowano nieco niektóre egzemplarze uzyskując pojazdy o różnej ilości miejsc dla pasażerów oraz różnych wskaźnikach spalania.

Ale ale, po co my o tym wspominamy... Otóż w czasie wzmożonego zainteresowania eksponatami marki "Żuk" znaleziono w jednym z nich manuskrypt pochodzący najpewniej z początku XXI wieku. Rekonstrukcja tekstu przytoczona jest poniżej:

> *Nie tak dawno temu w niedalekiej okolicy wykryto mocno podwyższony poziom Zatrważająco Złej Energii (ZZE). Globalna Grupa Porządkowa (GGP) podjęła zakrojone na szeroką skalę działania mające na celu znalezienie przyczyny tego zjawiska. Po długich i wyczerpujących poszukiwaniach znaleziono wreszcie źródło wspomnianej ZZE. Było to kilka niewielkich Zuchwale Zawiniętych Jam (ZZJ) zawierających pewną liczbę Nośników Negatywnej Energii (NNE). Podjęto badania nad sposobem ich usunięcia, a w efekcie przywrócenia ładu i harmonii na świecie.*
>
> *Okazało się, że jest tylko jeden sposób, aby zlikwidować ZZE. Istnieje bowiem pewien gatunek, który potrafi jednym dotknięciem zneutralizować NNE. Mowa o znanych powszechnie żukoskoczkach. Stworzonka te są niezwykle pożyteczne, mają jednak pewną zasadniczą...*

... i w tym miejscu tekst się urywał. Żukoskoczki były niezwykle dumne z tego, że ich przodkowie w ewolucji byli tak wyjątkowi i niepowtarzalni. Zastanawiające jednak, dlaczego owa planeta nie jest już taka błękitna... Czyżby nie udało się jej uratować przed zagładą? Czyżby Zatrważająco Zła Energia była

aż tak zatrważająco zła? I dlaczego akurat w tamtym okresie miała miejsca hiper-mutacja gatunku, która dała początek obecnej cywilizacji żukoskoczków?

Zostawmy jednak takie refleksje filozofom. Dość na tym, że od niedawna hitem na żukoskoczkowym rynku gier komputerowych stała się produkcja o tajemniczej nazwie "Snejk" czerpiąca fabułę z tego manuskryptu. Oczywiście nikt nie wie jak wyglądały owe Zuchwale Zawinięte Jamy i czy tylko ich Zawinięcie było zuchwałe, czy może same Jamy też...? Chociaż trochę...? We wspomnianej grze przyjęto, że taka Jama to prostopadłościenny kawałek przestrzeni (trójwymiarowej) o takiej właściwości (mówiąc w uproszczeniu), że jeśli dojdziemy do jednej ze ścian i pójdziemy jeszcze dalej, to pojawimy się po drugiej stronie. Nie potrafiono sobie tez wyobrazić w jaki sposób przodkowie żukoskoczków "neutralizowali" Nośniki Negatywnej Energii. Przyjęto więc, że po prostu je zjadały, wydłużając się nieco.

Gra jest prosta i grają w nią nawet dzieci. Porwała ostatnio również programistów z wszelakich firm. Z Twojej również. Zorganizowano nawet turniej w tą grę. O! Właśnie niedawno się zaczął! Zostały jeszcze niecałe 24 godziny... Zasady opisane są w dalszej części.

# Rozdział 2

# Komunikacja

Uzyskiwanie aktualnych informacji o wirtualnym świecie oraz wydawanie rozkazów jest możliwe za pomocą protokołu TCP/IP. Drużyna łączy się jako klient do odpowiedniego serwera konkursowego. Adres IP oraz port z którym należy się połączyć są podane w tabeli 5.2. Można nawiązać wiele połączeń jednocześnie, jednak sumaryczny transfer przypadający na każdy komputer jest ograniczony. Maksymalna ilość połączeń i maksymalny transfer podane są w "Ustaleniach Technicznych". Komunikacja odbywa się w trybie tekstowym. Bezpośrednio po połączeniu należy się zalogować, następnie sesja przechodzi w tryb poleceń.

## 2.1 Logowanie

Bezpośrednio po nawiązaniu połączenia serwer wysyła prośbę o login zakończoną znakiem końca linii: `LOGIN`. Należy wysłać swój login a następnie znak końca linii. Następnie serwer zapyta o hasło (`PASS`), na co należy analogicznie odpowiedzieć hasłem. Jeśli autoryzacja przebiegła pomyślnie serwer odpowie ciągiem znaków: `OK` i przejdzie w stan oczekiwania na komendy. W przeciwnym wypadku dostaniemy odpowiedź `FAILED 1 bad login or password` po czym nastąpi zamknięcie połączenia.

Poniżej znajduje się przykładowy zapis komunikacji w czasie logowania.

| klient → serwer | serwer → klient |
|---|---|
| | `LOGIN` |
| `login1` | |
| | `PASS` |
| `secret` | |
| | `OK` |

## 2.2 Tryb poleceń

Każde polecenie składa się z nazwy komendy (patrz niżej), argumentów (ilość zależna od polecenia) oraz znaku końca linii. Parametry powinny być oddzielone co najmniej jednym białym znakiem.

Na każdą komendę serwer odpowiada jednym z poniższych ciągów:

- '`OK`' — w przypadku zaakceptowania komendy
- '`FAILED` $e$ $msg$' — w przypadku błędu; gdzie $e$ to kod błędu, a $msg$ — komunikat błędu.

Następnie zależnie od komendy serwer może opcjonalnie wysłać lub odebrać dodatkowe dane. Jeśli dodatkowe dane są wysyłane od klienta do serwera, to po odebraniu tych danych serwer ponownie odpowie w sposób opisany powyżej.

Poniżej znajduje się przykładowy zapis komunikacji z serwerem zadania “Snejk”. Szczegółowy opis komend dla poszczególnych zadań znajdują się w podrozdziałach 3.3 i 4.2.

| klient → serwer | serwer → klient |
|---|---|
| | `LOGIN` |
| `login1` | |
| | `PASS` |
| `secret` | |
| | `OK` |
| `ME` | |
| | `OK` |
| | `0 7 1 1` |
| `CARRIERS` | |
| | `OK` |
| | `2` |
| | `0 3 2 9` |
| | `5 7 1 4` |
| `LOOK 0 7 1` | |
| | `OK` |
| | `...` |
| | `...` |
| | `...` |
| | |
| | `...` |
| | `.x.` |
| | `...` |
| | |
| | `...` |
| | `...` |
| | `...` |
| | |
| `GO` | |
| | `FAILED 3 bad format` |
| `GO 1` | |
| | `OK` |
| `WAIT` | |
| | `OK` |
| | `WAITING 0.599440` |
| | `OK` |
| `ME` | |
| | `OK` |
| | `1 7 1 1` |
| `LOOK -1 5 3` | |
| | `FAILED 102 bad point` |
| `SWITCH -1` | |
| | `FAILED 100 bad snake identifier` |

## 2.3 Konwencje

Jeśli nie jest zaznaczone inaczej, to przyjmujemy że:

- Każda linia zakończona jest pojedynczym znakiem o kodzie ASCII 20 (’\n’). Znak powrotu karetki (’\r’; kod 13) towarzyszący mu na niektórych systemach operacyjnych będzie traktowany jako biały znak.
- W przypadku danych przesyłanych przez serwer, liczby oraz wyrazy oddzielone są pojedynczą spacją.

- W przypadku danych przesyłanych od klienta do serwera (np. parametry komendy) dozwolona jest dowolna (niezerowa) ilość białych znaków pomiędzy danymi, a także na końcu oraz na początku linii.
- Za białe znaki uznajemy: spację, powrót karetki ('\r') oraz tabulator ('\t').

# Rozdział 3

# Przewóz

## 3.1 Opis problemu

### 3.1.1 Miasta

Jak wspomniano wcześniej, żukoskoczki stworzyły już kilka kolonii na różnych planetach. Każda kolonia składa się z pewnej ilości miast o pomijalnie małym promieniu w porównaniu z odległościami między nimi. Każde miasto można więc przedstawić jako punkt na płaszczyźnie. Żukoskoczki nie lubią wymyślać nazw, więc wszystko co się da numerują liczbami naturalnymi. Dotyczy to również miast.

Trzeba wspomnieć, że miasta na zasiedlanych planetach różnią się między sobą liczebnością mieszkańców. Ponieważ szansa na to, że dany żukoskoczek będzie chciał podróżować jest dla wszystkich taka sama, to w efekcie prawdopodobieństwo pojawienia się żukoskoczka oczekującego na "Żuka" w danym mieście jest proporcjonalna do zaludnienia tego miasta. Również szansa wyboru danego miasta jako docelowego jest proporcjonalna do liczby jego mieszkańców. Problem w tym, że nikt nigdy nie policzył ile żukoskoczków mieszka w poszczególnych miastach. Można jednak przyjąć, że w każdym jest ich tak wiele, że przemieszczenie się żukoskoczka z jednego miasta do innego nie zmienia zauważalnie szansy wyboru tego miasta.

### 3.1.2 Drogi

Żukoskoczki nigdy nie były biegłe w sporządzaniu map. Nie wiedzą tak naprawdę jakie drogi znajdują się na zasiedlanych planetach, które miasta łączą itp. Zaznaczanie zakrętów i skrzyżowań wykracza już daleko poza ich umiejętności. Drogi budowane są więc w taki sposób, że każda z nich łączy bezpośrednio w linii prostej dwa miasta i nie posiada skrzyżowań ani rozwidleń na całej długości. Kiedy jedna droga musiałaby przeciąć inną, budowany jest wiadukt, lub kopany tunel. Ponieważ drogi są proste, to ich długość jest zawsze równa odległości (euklidesowej) pomiędzy miastami które łączy. Żukoskoczki nie zwykły także budować dróg jednokierunkowych — każdą drogą można więc jechać w obu kierunkach.

Chociaż nie istnieją mapy dróg, to w każdym mieście ustawiane są drogowskazy mówiące do jakiego miasta prowadzi dana droga. Bywa czasem tak, że z jednego miasta do innego prowadzi kilka dróg. Zdarza się, że są to drogi zupełnie identyczne, jednak zwykle różnią się pewnymi parametrami. Jednym z takich parametrów jest opłata za wjazd na drogę. Wysokość tej opłaty jest dowolną nieujemną liczbą rzeczywistą i jest niezależna od tego w którą stronę jedziemy drogą.

Drogi różnią się między sobą również ograniczeniem szybkości. To dość oczywiste — im lepszej jakości droga, tym szybciej można się po niej poruszać. Mniej oczywisty jest fakt, że żukoskoczki nie uznają wyprzedzania, toteż aby uniknąć tworzenia się zatorów — wszystkie pojazdy muszą poruszać się z identyczną (maksymalną dopuszczalną) szybkością! W praktyce oznacza to, że czas przejazdu daną drogą jest zawsze taki sam i zależy tylko od jej długości i dozwolonej szybkości.

### 3.1.3 Budowa dróg

Co prawda większość kolonii jest pokryta gęstą siecią dróg, zdarzają się jednak i takie gdzie dróg jest niezbyt dużo. Bywa też, że drogi są za drogie (jak sama nazwa wskazuje), kiepskiej jakości (a więc przejazd nimi jest czasochłonny), albo po prostu nie ma takiej, jaka akurat jest potrzebna. Wówczas wystarczy zamówić budowę drogi i sprawa rozwiązana! No, prawie rozwiązana... Zwykle trzeba trochę poczekać, aż budowa drogi zostanie ukończona. Oczywiście nic nie jest za darmo — budowa dróg kosztuje i to niemało. Pocieszający jest jednak fakt, że po ukończeniu budowy takiej prywatnej drogi przechodzi ona na własność zamawiającego i wjazd na nią nic nie kosztuje. Co więcej — właściciel może sam ustalać opłaty za wjazd, jakie muszą uiścić inne firmy w momencie wjazdu na taką drogę. Wszystkie pieniądze z takich opłat trafiają do kieszeni właściciela. Czysty zysk!

### 3.1.4 Pojazdy

Każda drużyna ma do dyspozycji identyczny zestaw pojazdów (wszystkie marki "Żuk" oczywiście). Zestaw dostępnych żuków może być inny na każdej z planet. Żuki posiadają dwa istotne parametry. Jednym z nich jest ilość miejsc dla pasażerów (pojemność). Drugim natomiast koszt wozokilometra (czyli koszt przejechania jednego kilometra danym wozem).

Początkowo wszystkie pojazdy znajdują się w stolicy — mieście o numerze 0. Kiedy tylko właściciel uzna za stosowne, może kazać konkretnemu Żukowi wyjechać z miasta podaną drogą. Jest tylko jedno ograniczenie — pomiędzy wjechaniem do miasta, a wyjechaniem z niego musi upłynąć pewien czas. No cóż... korki. Jednak już od momentu wjechania można dowiedzieć się jakie drogi wychodzą z miasta, w którym wóz się aktualnie znajduje, oraz powiadomić kierowcę którą drogą powinien jechać dalej. Żuk wyjedzie z miasta tą drogą kiedy tylko będzie to możliwe. Oczywiście firma musi mieć dość pieniędzy (na danej planecie), ażeby opłacić przejazd tą drogą. Dodatkowo na bieżąco w miarę przebywanego dystansu będą pobierane pieniądze za każdy wozokilometr (jeśli braknie środków, to firma popadnie w długi i jej saldo będzie ujemne!). Tak więc w sumie przejazd daną drogą kosztować będzie: *[opłata za wjazd] + [długość drogi] · [koszt wozokilometra]*.

### 3.1.5 Podróżni

Kiedy żukoskoczek chce podróżować dzwoni do Centralnego Ośrodka Zamawiania Przewozów. Żukoskoczek podaje numer miasta, w którym się znajduje oraz miasto docelowe, do którego chciałby dotrzeć. Podaje też ile jest w stanie za ten przewóz zapłacić. Żukoskoczki są jednak tak samo rozrzutne jak niecierpliwe. Kiedy musi czekać, nieustannie podwyższa swoją ofertę w nadziei, że wreszcie ktoś skuszony wysokim zarobkiem przyjedzie po niego. Naturalnie przyrost oferowanej zapłaty za przejazd jest proporcjonalny do obecnie oferowanej kwoty. Współczynnik niecierpliwości, definiowany jako iloraz przyrostu zapłaty w jednostce czasu do obecnie oferowanej, jest taki sam dla wszystkich żukoskoczków na danej planecie. Co prawda nikt go w oficjalnych statystykach nigdy nie podał, to jednak nietrudno go zmierzyć.

Żukoskoczki oferują więc coraz to wyższe wynagrodzenie za przewóz, jednak i one mają ograniczone zasoby. Każdy żukoskoczek po pewnym czasie przestaje podnosić swoją ofertę i pozostaje ona stała aż do momentu, kiedy ktoś się ulituje i go przewiezie.

Sposób ustalania początkowej oferty nie jest dokładnie znany, wiadomo jednak że żukoskoczki kierują się tym ile musiały zaoferować inne zanim ktoś zechciał je przewieźć, oraz jak daleko (w linii prostej) chcą dojechać i jak daleko jechały tamte.

Ciekawostką jest fakt, że kiedy jeden żukoskoczek wsiądzie do Żuka, natychmiast pojawia się inny chętny do podróżowania. Przewozy stają się coraz popularniejsze, więc z czasem chętnych będzie coraz więcej.

Za niewygórowaną opłatą Centralny Ośrodek Zamawiania Przewozów chętnie udziela firmom przewozowym informacji na temat żukoskoczków oczekujących na transport. Należy jednak pamiętać, że nawet po krótkiej chwili informacja ta może być nieaktualna. Konkurencja nie śpi (no, chyba że jednak...).

W ramach programu rządowego "darmozjad", każdej firmie wypłacane są na bieżąco pieniądze mające zachęcić ją do rozwoju sieci transportowej.

Każda drużyna ma dostęp jednocześnie do kilku kolonii (tabela 5.2). Każda z kolonii jest wspólna dla wszystkich drużyn. Kolonie są zupełnie niezależne od siebie. Stałe dla każdej z nich znajdują się w tabeli 3.1.

## 3.2 Informacje szczegółowe

### Numeracja i jednostki

Miasta, drogi i żukoskoczki numerowane są liczbami naturalnymi (łącznie z 0). Numer jest jednocześnie identyfikatorem danego obiektu. Nie jest zagwarantowane, że numeracja będzie ciągła (tzn. może istnieć np. żukoskoczek o numerze 10 a nie istnieć taki o numerze 9). Dla każdego rodzaju obiektów jest osobna przestrzeń numeracji (tzn. w ramach jednej kolonii może istnieć miasto o numerze 10 i droga o numerze 10, ale nie mogą istnieć dwie drogi o takim samym numerze).

Jeśli nie podano inaczej, jednostką czasu jest 1 sekunda, jednostką odległości (długości) 1 km, a szybkości 1 km/s.

### Budowa drogi

Koszt budowy drogi wyrażony jest wzorem:

$$k = l \cdot s \cdot k_b$$

gdzie:

- $k$ — całkowity koszt
- $l$ — długość planowanej drogi
- $s$ — szybkość na drodze (minimalna i maksymalna wartość w tabeli 3.1)
- $k_b$ — współczynnik kosztu budowy drogi (tabela 3.1)

Czas budowy drogi wyraża się wzorem:

$$t = l \cdot t_b$$

gdzie:

- $t$ — całkowity czas budowy
- $l$ — długość planowanej drogi
- $t_b$ — współczynnik czasu budowy drogi (tabela 3.1)

### 3.2.1 Miasta i drogi

Miasta położone są na płaszczyźnie, a ich położenie określa jednoznacznie para liczb rzeczywistych $(x, y)$. Położenia miast są jawne. Przez cały czas konkursu miasta nie zmieniają się (pozycje są stałę, miasta nie znikają ani nie pojawiają się nowe).

Informacja o drogach wychodzących z miasta w którym znajduje się pojazd kierowany przez drużynę (wraz z szybkością jaka na niej obowiązuje i opłatą za wjazd) jest jawna. Drogi prywatne (należące do drużyn) nie są odróżniane od publicznych. Ilość dróg nie zmniejsza się, ale może się zwiększyć w przypadku wybudowania drogi przez którąś z drużyn. Szybkość i identyfikator drogi nie zmienia się w ciągu konkursu. Opłata za wjazd na drogę prywatną może zostać zmieniona przez właściciela, podczas gdy opłaty za wjazd na drogi państwowe nie zmieniają się.

### 3.2.2 Podróżni

Ilość żukoskoczków oczekujących na przewóz jest zależna od czasu. Pierwszy żukoskoczek pojawia się po czasie $p_f$ od rozpoczęcia konkursu, a kolejne w odstępach czasu $p_n$ od pojawienia się pierwszego. Dodatkowo — jak zostało napisane wcześniej — zawsze w momencie kiedy jakiś żukoskoczek wsiada do Żuka, pojawia się następny. Tak więc np. w przedziale czasu od $p_f$ do $p_f + p_n$ ciągle będzie oczekiwać na przejazd dokładnie jeden żukoskoczek (chociaż oczywiście — niekoniecznie ciągle ten sam), następnie dwa itd. Wartości stałych $p_n$ i $p_f$ podane są w tabeli 3.1.

Kiedy Żuk znajduje się w tym samym mieście, co żukoskoczek oczekujący na transport, drużyna może wydać odpowiednią komendę, która spowoduje zabranie żukoskoczka. Opłata za przewóz będzie równa kwocie oferowanej przez podróżnego w momencie wydania tej komendy i zostanie wpłacona do kasy firmy w momencie wjazdu do miasta docelowego. W tym też momencie żukoskoczek samodzielnie wysiądzie zwalniając miejsce w pojeździe. Czas przejazdu nie ma znaczenia. W Żuku może przebywać jednocześnie pewna ilość żukoskoczków. Informację o pojemności pojazdu można uzyskać używając odpowiedniej komendy.

### 3.2.3 Stałe

| Stała | Opis | A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|---|---|
| $p_f[\mathrm{h}]$ | czas, po jakim pojawia się pierwszy żukoskoczek | 0 | 0 | 0 | 0 | 0 |
| $p_n[\mathrm{h}]$ | odstępy czasu, po jakich pojawiają się kolejne żukoskoczki | 2 | $\infty$ | 2 | 2 | 2 |
| $k_b[\frac{\$}{\mathrm{km}\,\frac{\mathrm{km}}{\mathrm{s}}}]$ | koszt budowy 1 km drogi o szybkości 1 km/s | 100 | 500 | 100 | 100 | 100 |
| $t_b[\frac{\mathrm{s}}{\mathrm{km}}]$ | czas budowy 1 km drogi | 10 | 10 | 10 | 10 | 10 |
| $c_d[\frac{\$}{\mathrm{km}}]$ | ilość pieniędzy wpływających w ramach programu "darmozjad" w czasie jednej sekundy | 1 | 5 | 1 | 1 | 1 |
| $s_{min}[\frac{\mathrm{km}}{\mathrm{s}}]$ | minimalna szybkość na nowo budowanej drodze | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| $s_{max}[\frac{\mathrm{km}}{\mathrm{s}}]$ | maksymalna szybkość na nowo budowanej drodze | 5 | 5 | 5 | 5 | 20 |
| $t_s[\mathrm{s}]$ | czas przejazdu przez miasto (pomiędzy wjechaniem do miasta a wyjechaniem z niego) | 2 | 1 | 2 | 2 | 2 |
| $c_p[\$]$ | koszt jednego zapytania o listę oczekujących żukoskoczków | 1 | 1 | 20 | 1 | 1 |

Tabela 3.1: Stałe dla zadania "Przewóz" w różnych konkurencjach

## 3.3 Komendy

Ogólne założenia protokołu komunikacji (łączenie się, logowanie, wysyłanie komend) opisane są w rozdziale 2. Poniżej znajduje się lista komend dostępnych dla zadania "Przewóz".

**ME** Zwraca informacje o obecnie sterowanym Żuku.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia:

  - $X$ ($X \in \mathbb{N}$) — numer Żuka
  - $c$ ($c \in \mathbb{N}$) — pojemność (ilość pasażerów jaką Żuk może pomieścić)
  - $r$ ($r \in \mathbb{R}_+$) — spalanie
  - $p$ ($p \leqslant c$) — ilość pasażerów znajdujących się obecnie w Żuku
- Druga linia zależy od tego czy Żuk znajduje się w mieście czy na drodze z jednego miasta do drugiego:
  - W przypadku miasta:
    * wyraz `TOWN`
    * $T$ — miasto w którym Żuk się znajduje
    * $l$ — czas po jakim Żuk będzie mógł wyjechać z miasta
  - W przypadku drogi:
    * wyraz `ROAD`
    * $T_{src}$ — numer miasta z którego Żuk wyjechał
    * $T_{dst}$ — to numer miasta docelowego
    * $s$ — to szybkość jaka obowiązuje na drodze którą Żuk się porusza
    * $l$ — czas po jakim Żuk osiągnie miasto docelowe
- W kolejnych $p$ liniach znajdują się opisy pasażerów. Każda linia opisu składa się z następujących liczb:
  - $p$ ($p \in \mathbb{N}$) — numer (identyfikator) żukoskoczka
  - $T_{pdst}$ ($T_{pdst} \in \mathbb{N}$) — miasto docelowe, do którego zmierza pasażer (żukoskoczek)
  - $f$ ($f \in \mathbb{R}$) — zapłata jaką pasażer uiści po dostarczeniu go do wskazanego miasta

**CASH** Zwraca obecną ilość pieniędzy w kasie firmy.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza (i jedyna) linia:
  - $C$ — saldo (ilość pieniędzy w kasie firmy)
  - $i_d$ — przychód w ramach programu "darmozjad"
  - $i_c$ — przychód z opłat za przejazd (innych drużyn) drogami wybudowanymi przez Twoją drużynę
  - $i_p$ — przychód z opłat za przewóz
  - $o_f$ — koszty (ze znakiem "-") jazdy Żukami (dystans przejechany każdym z nich pomnożony przez koszt wozokilometra)
  - $o_c$ — suma opłat za wjazd na drogę (ze znakiem "-")
  - $o_b$ — koszty dróg zbudowanych przez drużynę (ze znakiem "-")
  - $o_p$ — kwota wydana na sprawdzania listy oczekujących żukoskoczków (ze znakiem "-")

**CARS** Zwraca informacje o dostępnych Żukach.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba dostępnych Żuków
- Kolejnych $2n$ linii zawiera opisy Żuków, każdy złożony z dwóch linii. Opis ten jest identyczny jak dwie pierwsze linie opisu w przypadku komendy **ME** (lista pasażerów jest pominięta)

**SWITCH** Wybiera Żuka, do którego będą odnosić się polecenia: `ME`, `WAYS`, `GO`, `WAIT`. Informację o dostępnych Żukach zwraca komenda `CARS`

**Parametry**:

- $X$ ($X \in \mathbb{N}$) — numer (identyfikator) Żuka

**TOWNS** Zwraca pozycje i numery miast znajdujących się na planecie.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba miast
- Każda z kolejnych $n$ linii:
  - $T$ ($T \in \mathbb{N}$) — numer miasta
  - $x$ ($x \in \mathbb{R}$) — współrzędna "x" miasta
  - $y$ ($y \in \mathbb{R}$) — współrzędna "y" miasta

**WAYS** Zwraca listę dróg wychodzących z miasta w którym Żuk się znajduje.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba dróg
- Każda z kolejnych $n$ linii:
  - $R$ ($R \in \mathbb{N}$) — numer drogi
  - $T$ ($T \in \mathbb{N}$) — miasto docelowe
  - $s$ ($s \in \mathbb{R}_+$) — szybkość jaka obowiązuje na drodze
  - $c$ ($c \in \mathbb{R}$) — opłata za wjazd na drogę (nie uwzględnia kosztu paliwa)
  - $l$ ($l \in \mathbb{R}_+$) — długość drogi

**GO** Wydaje Żukowi rozkaz wyjechania z miasta podaną drogą. Rozkaz jest poprawny tylko jeśli Żuk znajduje się w mieście. Drużyna musi mieć wystarczającą ilość pieniędzy, aby uiścić opłatę za wjazd na drogę. Jeśli Żuk nie może jeszcze wyjechać z miasta, to komenda ta zostanie zapamiętana i zrealizowana dopiero kiedy będzie to możliwe. Listę możliwych do użycia dróg zwraca komenda `WAYS`.

**Parametry**:

- $R$ ($R \in \mathbb{N}$) — numer drogi, którą Żuk powinien wyjechać

opcjonalny $c_{max}$ ($c \in \mathbb{R}_+$) — maksymalna wartość opłaty za wjazd na drogę, jeśli opłata byłaby wyższa, to komenda zakończy się błędem[1]

**LIST** Zwraca listę żukoskoczków czekających w różnych miastach na przewóz. Drużyna musi mieć wystarczająco dużo pieniędzy, aby zapłacić za tę usługę ($c_p$ w tabeli 3.1).

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba żukoskoczków oczekujących na przewóz
- Każda z kolejnych $n$ linii:

[1] Zauważmy, że opłata za wjazd na drogę może ulec zmianie od momentu wywołania komendy `WAYS` jeśli droga jest prywatna

- $P$ ($P \in \mathbb{N}$) — numer żukoskoczka (identyfikator)
- $T_{psrc}$ ($T_{psrc} \in \mathbb{N}$) — numer miasta w którym czeka
- $T_{pdst}$ ($T_{pdst} \in \mathbb{N}$) — numer miasta do którego chce dojechać
- $f$ ($f \in \mathbb{R}$) — aktualna zapłata za przewóz oferowana przez żukoskoczka

**TAKE** Wydaje Żukowi rozkaz zabrania żukoskoczka. Żuk musi znajdować się w tym samym mieście co żukoskoczek. W Żuku musi być co najmniej jedno miejsce wolne.

**Parametry**:

- $P$ ($P \in \mathbb{N}$) — numer (identyfikator) żukoskoczka

**MY_ROADS** Zwraca listę dróg wybudowanych przez drużynę.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba dróg, jakie zbudowała (lub zaczęła budować) drużyna, która wydaje to polecenie
- Każda z kolejnych $n$ linii:
  - $R$ ($R \in \mathbb{N}$) — numer drogi
  - $T_1$ ($T_1 \in \mathbb{N}$) — numer pierwszego z miast które łączy droga
  - $T_2$ ($T_2 \in \mathbb{N}$) — numer drugiego z miast które łączy droga
  - $s$ ($s \in \mathbb{R}_+$) — szybkość jaka obowiązuje na drodze
  - $c$ ($c \in \mathbb{R}$) — opłata za wjazd na drogę
  - $l$ ($l \in \mathbb{R}_+$) — długość drogi
  - $u$ ($u \in \mathbb{N}$) — ilość przejazdów drogą
  - $o$ ($o \in \mathbb{R}$) — dochód z drogi
  - $l$ ($l \in \mathbb{R}$) — czas pozostały do zbudowania drogi, lub 0 jeśli jest już zbudowana

**BUILD** Rozpoczyna budowę drogi. Drużyna musi dysponować odpowiednią ilością pieniędzy. Stan budowy drogi można sprawdzić za pomocą komendy `MY_ROADS`. Droga będzie widoczna za pomocą komendy `WAYS` dopiero po ukończeniu budowy.

**Parametry**:

- $T_1$ ($T_1 \in \mathbb{N}$) — numer pierwszego z miast które ma łączyć droga
- $T_2$ ($T_2 \in \mathbb{N}$) — numer drugiego z miast które ma łączyć droga
- $s$ ($s \in \mathbb{R}_+$) — szybkość jaka ma obowiązywać na drodze
- $c$ ($c \in \mathbb{R}$) — opłata za wjazd na drogę dla innych drużyn

**ALTER** Zmienia opłatę za wjazd na drogę prywatną. Drużyna wydająca komendą musi być jej właścicielem. Można zmienić opłatę drogi, która jest jeszcze w budowie. Listę dróg drużyny zwraca komenda `MY_ROADS`

**Parametry**:

- $R$ ($R \in \mathbb{N}$) — numer drogi
- $c$ ($c \in \mathbb{R}$) — nowa opłata dla innych drużyn za wjazd na drogę

**TIME** Zwraca czas jaki upłynął od rozpoczęcia gry.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza (i jedyna) linia:
  - $t$ ($t \in \mathbb{R}$) — czas w sekundach
  - $HH$:$MM$:$SS$.$mmm$ — czas w podanym formacie ($HH$ — godziny, $MM$ — minuty, $SS$ — sekundy, $mmm$ — milisekundy

**WAIT** Czeka do momentu wyjazdu z miasta lub przyjazdu do miasta docelowego, po czym zwraca kontrolę. Do tego momentu odbieranie komend od klienta jest wstrzymane. W momencie zwrócenia kontroli serwer wysyła informację o tym. Możliwe jest też podanie maksymalnego czasu oczekiwania (jeśli oczekiwanie byłoby dłuższe, to kontrola zostanie zwrócona po tym czasie).

**Parametry**:

opcjonalny $t$ ($t \in \mathbb{R}$) — maksymalny czas oczekiwania; czas oczekiwania będzie nie większy niż $t$

**Dane (od serwera)**:

- Pierwsza linia (wysyłana natychmiastowo):
  - wyraz `WAITING`
  - $t$ ($t \in \mathbb{R}$) — czas w sekundach, po jakim zostanie zwrócona kontrola
- Druga linia (wysyłana tuż przed zwróceniem kontroli):
  - wyraz `OK`

## 3.4 Punktacja

Za każdego żukoskoczka dowiezionego do celu drużyna dostaje jeden punkt.

# Rozdział 4

# Snejk

## 4.1 Opis problemu

Aby nie mylić żukoskoczków w obecnym stadium cywilizacji, z postaciami w grze "Snejk" stworzonymi (wg. autorów gry) na wzór przodków żukoskoczków, będziemy postacie z gry nazywać "snejkami".

### 4.1.1 Zuchwale Zawinięte Jamy

Zasady gry są proste. W Zuchwale Zawiniętej Jamie znajdują się snejki oraz Nośniki Negatywnej Energii. Jamy mają kształt prostopadłościanu o wymiarach $W \times H \times D$ wyrażonych liczbami całkowitymi. Jamę można więc podzielić się na $W \cdot H \cdot D$ jednostkowych sześcianów nie zachodzących na siebie. Dla uproszczenia, będziemy je nazywać polami. Położenie (pozycja) obiektu jest wyrażona liczbami całkowitymi $(x, y, z)$ takimi że $0 \leqslant x < W$, $0 \leqslant y < H$, $0 \leqslant z < D$. Pozycja taka oznacza, że obiekt zajmuje sześcian posiadający wierzchołki w punktach $(x, y, z)$, $(x + 1, y + 1, z + 1)$.

Każda drużyna ma dostęp do kilku Zuchwale Zawiniętych Jam. Jamy te są wspólne dla wszystkich drużyn. W każdej z nich drużyna posiada od jednego do kilku snejków (stała $N_s$ w tabeli 4.1)

### 4.1.2 Snejki i ruchy

Czas gry podzielony jest na równe tury o czasie trwania $t_t$ (stała w tabeli 4.1). W czasie każdej z nich można wydawać rozkazy snejkom. Na koniec tury snejki poruszają się zgodnie z rozkazem lub zostają w miejscu. Możliwy jest ruch tylko o jednostkową odległość wzdłuż jednego z trzech wymiarów w kierunku rosnącej lub malejącej współrzędnej (np. ruch w kierunku rosnącej współrzędnej $x$).

Jak wspomniano wcześniej, jamy po których poruszają się snejki są zawinięte. Oznacza to, że np. kiedy głowa snejka znajduje się na polu $(W - 1, y, z)$ i poruszy się w kierunku rosnącej współrzędnej $x$, to znajdzie się na polu $(0, y, z)$. Podobnie kiedy jej współrzędne to $(0, y, z)$ i poruszy sie w kierunku malejącej współrzędniej $x$ — wówczas przemieści się na $(W - 1, y, z)$. Analogicznie jest w przypadku innych współrzędnych.

Snejki składają się z jednej głowy i dowolnej ilości członów. Ruch snejka polega na przemieszczeniu się jego głowy w kierunku zgodnym z rozkazem oraz przesunięciu się pierwszego członu w miejsce głowy, drugiego w miejsce pierwszego itd.

Na początku każdy snejk posiada tylko głowę. W momencie, kiedy głowa wejdzie na pole zajmowane przez Nośnik Negatywnej Energii, to snejk zjada ten nośnik wydłużając się o jeden człon [1] (człon ten pojawi się w miejscu, gdzie był przed chwilą ostatni człon, lub — jeśli snejk nie posiadał członów — tam gdzie głowa).

[1] z wyjątkiem przypadku, kiedy ciała snejków zajmują 80% lub więcej całkowitego miejsca na w jamie

Niestety wszystkie pracowicie "zebrane" człony można stracić w jednym momencie! Dzieje się tak, kiedy snejk umiera. Na szczęście w tym samym momencie odradza się (w losowym miejscu), jednak wszystkie człony znikają. Snejk umrze, jeśli w momencie zakończenia tury wejdzie na pole, na którym:

- w czasie trwającej tury znajdowała się głowa albo człon dowolnego snejka lub
- w kolejnej turze będzie znajdowała się głowa lub człon innego snejka lub
- w kolejnej turze znajdowałaby się głowa innego snejka, gdyby ten nie umarł

Zauważmy, że kiedy kilka snejków wejdzie na to samo pole w jednym momencie, to ginie każdy z nich.

### Nośniki Negatywnej Energii

Każdy nośnik zajmuje dokładnie jeden sześcian jednostkowy. Pozycje wszystkich nośników są jawne i można je odczytać za pomocą odpowiedniej komendy. Ilość nośników zależy tylko od czasu. Pierwszy pojawia się po czasie $n_f$ od rozpoczęcia konkursu, a kolejne w odstępach czasu $n_n$ od pojawienia się pierwszego. W momencie kiedy jeden zostanie zjedzony, natychmiast pojawia się następny, tak więc w tym wypadku ich ilość nie zmienia się. Wartości stałych $n_n$ i $n_f$ podane są w tabeli 4.1.

Jeśli Nośnik Negatywnej Energii nie zostanie zjedzony przez pewną ilość tur od pojawienia się ($t_n$), to znika i pojawia sie w innym miejscu.

### Pozyskiwanie informacji

Snejki, mimo że są ślepe, posiadają zdolności parapsychiczne. Potrafią:

1. w dowolnym momencie sprawdzić co znajduje się na wybranym polu oraz w jego otoczeniu
2. cofnąć się w przeszłość i sprawdzić gdzie przesunęły się, albo pojawiły głowy snejków na koniec podanej tury (maksymalnie 1800 tur wstecz)

Takie działania wymagają jednak dużo skupienia i sa wyczerpujące. W grze operujemy pojęciem "punkty PSI". Co turę, do dyspozycji drużyny trafiają 2 punkty PSI (w sumie można ich uzbierać max 300). Obie wspomniane wyżej akcje zużywają po 1 punkcie PSI.

Umiejętności snejków pozwalają też sprawdzać gdzie znajdują się Nośniki Negatywnej Energii, te jednak są wyczuwalne bez większego wysiłku (ze względu na duże natężenie Zatrważająco Złej Energii), a więc nie wymagają zużycia cennych punktów PSI.

## 4.1.3 Stałe

| Stała | Opis | B1 | B2 | B3 |
|---|---|---|---|---|
| $t_t$[s] | czas trwania tury | | 1 | |
| $t_n$[tur] | czas, po jakim nośnik odnawia się w innym miejscu | 120 | 180 | 120 |
| $n_f$[h] | czas, po jakim pojawia się pierwszy nośnik | 1 | 1 | 4 |
| $n_n$[h] | odstępy czasu, w jakich pojawiają się kolejne nośniki | 4 | $\infty$ | 4 |
| $N_s$ | ilość snejków przypadających na jedną drużynę | 1 | 1 | 3 |
| $p_c$ | bonus za każdy człon w momencie zjedzenia nośnika | 0.1 | 0.5 | 0.2 |
| $W \times H \times D$ | wymiary planszy | $20 \times 20 \times 20$ | $20 \times 20 \times 20$ | $40 \times 40 \times 40$ |

Tabela 4.1: Stałe dla zadania "Snejk" w różnych konkurencjach

## 4.2 Komendy

Ogólne założenia protokołu komunikacji (łączenie się, logowanie, wysyłanie komend) opisane są w rozdziale 2. Poniżej znajduje się lista komend dostępnych dla zadania "Snejk".

**ME** Zwraca informacje o obecnie sterowanym snejku.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia:
  - $x$ ($x \in \mathbb{N}$, $0 \leqslant x < W$) — współrzędna $x$ głowy
  - $y$ ($y \in \mathbb{N}$, $0 \leqslant y < H$) — współrzędna $y$ głowy
  - $z$ ($z \in \mathbb{N}$, $0 \leqslant z < D$) — współrzędna $z$ głowy
  - $l$ ($l \in \mathbb{N}$) — długość snejka (głowa + człony)

**SWITCH** Wybiera snejka, do którego będą odnosić się polecenia: `ME` i `GO`.

**Parametry**:

- $S$ ($S \in \mathbb{N}$, $0 \leqslant S < N_s$) — numer (identyfikator) snejka

**CARRIERS** Zwraca informacje o położeniach Nośników Negatywnej Energii.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza linia: $n$ — liczba Nośników Negatywnej Energii znajdujących się w jamie w bieżącej turze
- Każda z kolejnych $n$ linii:
  - $x$ ($x \in \mathbb{N}$, $0 \leqslant x < W$) — współrzędna $x$ nośnika
  - $y$ ($y \in \mathbb{N}$, $0 \leqslant y < H$) — współrzędna $y$ nośnika
  - $z$ ($z \in \mathbb{N}$, $0 \leqslant z < D$) — współrzędna $z$ nośnika
  - $l$ ($l \in \mathbb{N}$, $1 \leqslant l < t_n$) — ilość tur po których nośnik zniknie (licząc bieżącą)

**CREDITS** Zwraca obecną ilości punktów PSI

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza (i jedyna) linia:
  - $s$ ($s \in \mathbb{N}$) — ilość punktów PSI

**DIMENSIONS** Zwraca wymiary Zuchwale Zawiniętej Jamy

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza (i jedyna) linia:
  - $W$ ($W \in \mathbb{N}$) — rozciągłość $x$ jamy
  - $H$ ($H \in \mathbb{N}$) — rozciągłość $y$ jamy
  - $D$ ($D \in \mathbb{N}$) — rozciągłość $z$ jamy

**TURN** Zwraca numer trwającej obecnie tury.

**Parametry**:brak
**Dane (od serwera)**:

- Pierwsza (i jedyna) linia:
  - $t$ ($t \in \mathbb{N}$) — numer aktualnie trwającej tury

**GO** Decyduje jak snejk ma się poruszyć na koniec tury. Jeśli komenda będzie wywołana kilkakrotnie dla jednego snejka w czasie jednej tury, to decyduje ostatnia.

**Parametry**:

- $m$ ($m \in \{-3, -2, -1, 0, 1, 2, 3\}$) — instrukcja ruchu oznaczająca:
  - 0 — brak ruchu (może posłużyć do anulowania wcześniejszych)
  - 1 — ruch w kierunku rosnącej współrzędnej $x$
  - 2 — ruch w kierunku rosnącej współrzędnej $y$
  - 3 — ruch w kierunku rosnącej współrzędnej $z$
  - $-1$ — ruch w kierunku malejącej współrzędnej $x$
  - $-2$ — ruch w kierunku malejącej współrzędnej $y$
  - $-3$ — ruch w kierunku malejącej współrzędnej $z$

**LOOK** Sprawdza co znajduje się na podanym polu i jego otoczeniu. W efekcie dostajemy informację o głowach/ciałach snejków oraz Nośnikach Negatywnej Energii w sześcianie o wymiarach $3 \times 3 \times 3$. Komenda zużywa jeden punkt PSI.

**Parametry**:

- $x_0$ ($x_0 \in \mathbb{N}$, $0 \leqslant x_0 < W$) — współrzędna $x$ punktu
- $y_0$ ($y_0 \in \mathbb{N}$, $0 \leqslant y_0 < H$) — współrzędna $y$ punktu
- $z_0$ ($z_0 \in \mathbb{N}$, $0 \leqslant z_0 < D$) — współrzędna $z$ punktu

**Dane (od serwera)**:

- 3 razy podany niżej zestaw (kolejno dla $z = z_0 - 1$, $z = z_0$, $z = z_0 + 1$)
  - 3 linie (kolejno dla $y = y_0 - 1$, $y = y_0$, $y = y_0 + 1$), każda po 3 znaki (kolejno dla $x = x_0 - 1$, $x = x_0$, $x = x_0 + 1$), oznaczające:
    - `.` (kropka) — pole jest puste
    - `x` — pole zawiera ciało snejka (głowę lub człon)
    - `o` — na polu znajduje się Nośnik Negatywnej Energii

**CHANGES** Sprawdza gdzie przesunęły się lub pojawiły głowy snejków na koniec podanej tury oraz gdzie pojawiły się Nośniki Negatywnej Energii. Lista zawiera tylko różnice pomiędzy turą $t + 1$ a $t$, tak więc głowy snejków które sie nie poruszyły nie będą uwzględnione w wykazie. Komenda zużywa jeden punkt PSI.

**Parametry**:

- $t$ ($t \in \mathbb{N}$) — numer tury

**Dane (od serwera)**:

- Pierwsza linia: $n_1$ — liczba snejków których głowy przemieściły się

- Każda z kolejnych $n_1$ linii zawiera współrzędne pola na którym pojawiła się głowa jakiegoś snejka:
  - $x$ ($x \in \mathbb{N}$, $0 \leqslant x < W$) — współrzędna $x$
  - $y$ ($y \in \mathbb{N}$, $0 \leqslant y < H$) — współrzędna $y$
  - $z$ ($z \in \mathbb{N}$, $0 \leqslant z < D$) — współrzędna $z$
- $n_1+2$-ta linia: $n_2$ — liczba Nośników Negatywnej Energii, które pojawiły się na koniec podanej tury
- Każda z kolejnych $n_2$ linii zawiera współrzędne pola na którym pojawił się nośnik:
  - $x$ ($x \in \mathbb{N}$, $0 \leqslant x < W$) — współrzędna $x$
  - $y$ ($y \in \mathbb{N}$, $0 \leqslant y < H$) — współrzędna $y$
  - $z$ ($z \in \mathbb{N}$, $0 \leqslant z < D$) — współrzędna $z$

**WAIT** Czeka do końca podanej tury. Jeśli tura już się skończyła, to czas oczekiwania wynosi 0. W czasie oczekiwania odbieranie komend od klienta jest wstrzymane. W momencie zwrócenia kontroli serwer wysyła informację o tym. Możliwe jest też podanie maksymalnego czasu oczekiwania (jeśli oczekiwanie byłoby dłuższe, to kontrola zostanie zwrócona po tym czasie).

**Parametry**:

- $t$ ($t \in \mathbb{N}$) — numer tury

**Dane (od serwera)**:

- Pierwsza linia (wysyłana natychmiastowo):
  - wyraz `WAITING`
  - $t$ ($t \in \mathbb{R}$) — czas w sekundach, po jakim zostanie zwrócona kontrola
- Druga linia (wysyłana tuż przed zwróceniem kontroli):
  - wyraz `OK`

## 4.3 Punktacja

Za każdy zneutralizowany Nośnik Negatywnej Energii drużyna dostaje 1 punkt oraz bonus za ilość członów snejka który tego dokonał w wysokości $p_c$ (tabela 4.1) punktów za każdy z członów, jakie posiadał przed zjedzeniem.

# Rozdział 5

# Rodzaje konkurencji i punktacja końcowa

## 5.1 Punkty rankingowe

W każdej konkurencji drużyny zdobywają punkty. Nie są one jednak bezpośrednio liczone do rankingu końcowego. Zamiast tego liczone są tzw. **punkty rankingowe**. Są one wyznaczane dla każdej drużyny jako iloczyn liczby 100 oraz stosunku liczby punktów zdobytych przez tą drużynę w danej konkurencji do średniej z ilości punktów trzech najwyżej ocenionych (w danej konkurencji) graczy.

$$p_r(d) = 100 \cdot \frac{p(d) \cdot 3}{p(d_1) + p(d_2) + p(d_3)}$$

gdzie:

- $p_r(d)$ — liczba punktów rankingowych drużyny $d$ za daną konkurencję
- $p(d)$ — liczba punktów drużyny $d$ za daną konkurencję
- $d_1$, $d_2$, $d_3$ — trzy pierwsze drużyny z listy drużyn posortowanej malejąco ze względu na liczbę punktów w danej konkurencji

Przykład ilustruje tabela 5.1. O ostatecznym miejscu w rankingu decyduje suma punktów rankingowych zdobytych we wszystkich konkurencjach w każdym ze światów.

| Drużyna | Punkty za konkurencję x | Punkty rankingowe za konkurencję x |
|---|---|---|
| team3 | 22 | 110 |
| team5 | 22 | 110 |
| team4 | 16 | 80 |
| team1 | 15 | 75 |
| team2 | 10 | 50 |
| team6 | 5 | 25 |

Tabela 5.1: Przykładowa punktacja graczy

## 5.2 Sytuacje awaryjne

W razie wystąpienia sytuacji, w której nie wszystkie drużyny są w stanie brać udział w konkursie na ustalonych zasadach (np. brak prądu, awaria sieci lokalnej lub jej części, problemy z systemem konkursowym, etc.) oraz wina nie leży po stronie tych drużyn ani ich sprzętu, organizatorzy wstrzymają działanie systemu konkursowego, a o jego ponownym uruchomieniu uczestnicy zostaną poinformowani w lokalnym serwisie WWW konkursu. W tym czasie punkty nie będą naliczane. W takiej sytuacji mogą zostać zerwane wszystkie połączenia z serwerem konkursowym.

## 5.3 Serwery

| Zadanie | Zestaw (konkurencja) | Adres:port |
|---|---|---|
| Taxi | A1 | server.dl24:20000 |
| Taxi | A2 | server.dl24:20001 |
| Taxi | A3 | server.dl24:20002 |
| Taxi | A4 | server.dl24:20003 |
| Taxi | A5 | server.dl24:20004 |
| Snejk | B1 | server.dl24:20005 |
| Snejk | B2 | server.dl24:20006 |
| Snejk | B3 | server.dl24:20007 |

Tabela 5.2: Adresy i porty dla poszczególnych konkurencji