Dokumentacja wstępna
z projektu rozmieszczenie towarów (2D) w magazynie
z przedmiotu POP

Wykonali:
Denys Fokashchuk
Bartosz Bąbolewski

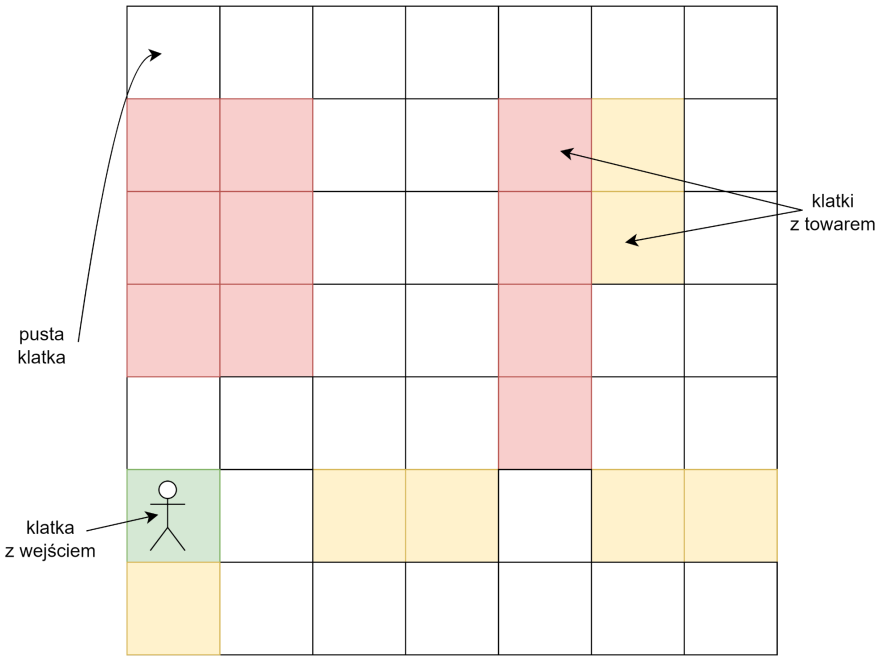
Opis zadania:

Stworzyć program, który za pomocą Algorytmu Ewolucyjnego rozwiąże problem rozmieszczenia towarów (2D) w magazynie. Interfejs graficzny programu powinien umożliwiać definiowanie różnych kształtów magazynu oraz towarów o różnej długości i szerokości. Między towarami powinna być zachowana przestrzeń potrzebna na swobodne przemieszczanie się magazyniera. Program powinien maksymalizować powierzchnię towarów w magazynie. Interfejs programu powinien umożliwiać graficzną prezentację wyniku.

Informacje elementarne

Nie jesteśmy do końca pewni jakie rozwiązanie pewnych problemów wybierzemy ostatecznie, dlatego w niektórych momentach zostaną podane kilka sugestii solucji problemów.

Magazyn reprezentowany jest w postaci siatki składającej się z klatek. Na rysunku 1. można zobaczyć przykładową ilustrację dopuszczalnego rozwiązania. Pokazano tam siatkę o rozmiarze 7x7 klatek. Każda klatka może być w jednym z trzech stanów: pusta, z towarem lub z wejściem.

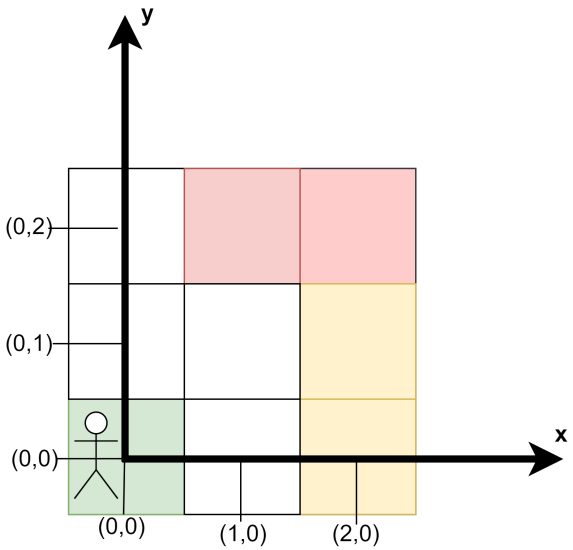


Rysunek 1. Przykładowe dopuszczalne rozmieszczenie towarów

Klatki z towarami będą miały kolory czerwony, żółty oraz ewentualnie inne w celu pokazania, że niektóre klatki (szczególnie te, które są obok siebie) reprezentują różne towary.

Jeszcze nie jesteśmy do końca pewni czy zrobimy taki algorytm, że klatka z wejściem będzie taka, którą poda użytkownik lub że będzie to dowolna klatka “brzegowa”. Pierwsze rozwiązanie ma sens wtedy, gdy jest już zbudowany magazyn i chcemy

tylko ulokować towary, natomiast drugie rozwiązanie jest sensowne wtedy, gdy jest jakiś pokój i wiemy jakimi rzeczami o jakich wymiarach chcemy ten pokój wypełnić (na przykład jeśli istnieje przestrzeń do budowania fabryki, wiadoma jest liczba oraz rozmiary urządzeń, które muszą tam być i na tej podstawie chcemy zrozumieć gdzie najlepiej zrobić wejście).
W celu ułatwienia wizualizacji będziemy używać rysunków, jednak dane zostaną użyte w postaci liczb, przy tym stworzymy układ współrzędnych, który pokazano na rysunku 2.



Rysunek 2. Układ współrzędnych

Każdy punkt jest środkiem każdej klatki, co ułatwia pokazanie w której klatce co się znajduje. Lokalizację towarów będziemy pokazywać za pomocą przedziałów dla poszczególnych osi. W powyższym przypadku współrzędne każdego obiektu są następujące:

1. wejście: (0,0)
2. lokalizacja czerwonego towaru: (1:2, 2:2).
3. lokalizacja żółtego towaru: (2:2, 0:1).

Reprezentacja osobnika

W ogólnym przypadku rozwiązanie (osobnik), będzie zawierał informacje lokalizacji towarów wraz z swoją oceną. Podjęliśmy decyzję o zawieraniu informacji o ocenie w celu mniejszej liczby obliczeń, bo ta informacja przyda się w różnych częściach algorytmu, jednak wiąże się to z nieco większą pamięcią, którą potrzebuje program. Informacja o lokalizacji n towarów ma postać:

```
[
        [x1_start, x1_end, y1_start, y1_end],
        [x2_start, x2_end, y2_start, y2_end],
        …
        [xn_start, xn_end, yn_start, yn_end]
],
```

gdzie

xk_start i *xk_end* to współrzędne początku i końca odpowiednie towaru k na osi x,
yk_start i *yk_end* to współrzędne początku i końca odpowiednie towaru k na osi y

Krótki opis naszego rozwiązania:

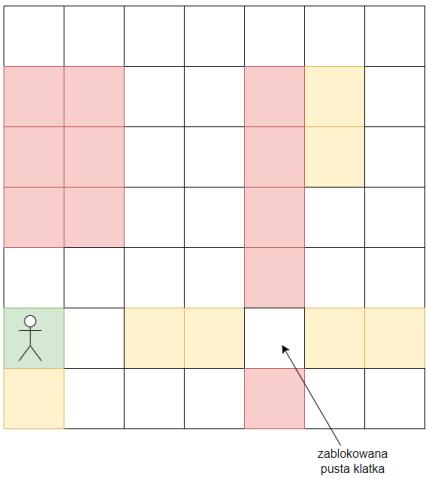
Zdecydowaliśmy “rozbić” naszą aplikację na dwa byty:

1. Główny program, który zawiera funkcje niezbędne do działania algorytmu ewolucyjnego, który zostanie omówiony dogłębniej w dalszej części dokumentacji wstępnej.
2. Aplikacja UI, która umożliwi zadawania parametrów magazynu, kształtu towarów oraz innych parametrów i także udostępniać możliwość komunikacji z głównym programem w celu otrzymania wyniku, tj. optymalnym rozmieszczeniem towarów.

Postanowiliśmy, że dopuszczalnym rozmieszczeniem układu towarów będzie takie rozmieszczenie, iż:

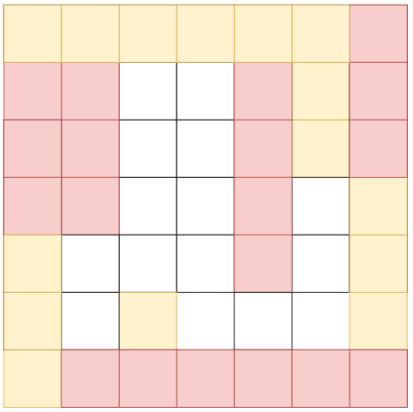
1. dostępna co najmniej jedna klatka “brzegowa” (ona sugeruje, iż jest dostęp do magazynu zewnątrz) albo wskazana przez użytkownika klatka brzegowa;
2. od dowolnej pustej klatki można dojść do innej dowolnej pustej klatki;
3. żadna klatka nie zawiera więcej niż jeden towar.

Przykładowe niedopuszczalne rozwiązanie ze względu na blokadę pustej klatki pokazano na rysunku 3.

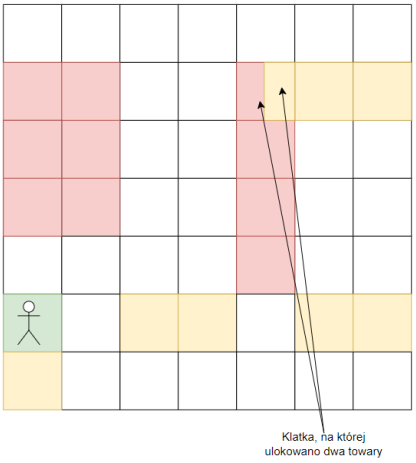


Rysunek 3. Przykładowe niedopuszczalne rozwiązanie

Jak wspomniano wyżej, niedopuszczalnym rozwiązaniem może być takie, że nie ma dostępu do wejścia do magazynu lub takie, że na jednej klatce ulokowano co najmniej dwa towary. Te sytuacje pokazano na rysunkach 4. oraz 5. odpowiednio.



Rysunek 4. Zablokowane wejście w magazynie



Rysunek 5. Rozmieszczenie dwóch towarów w jednej klatce

Opis głównego programu

Mutacje

Wybieramy losowo towar w genotypie, któremu będziemy zmieniać położenie. Mutacje będą powodowały powstawanie nowych osobników, w których towar może być przesunięty w osi x lub w osi y o pewną, losową wartość generowaną z rozkładu Gaussa.

Algorytm selekcji turniejowej

Wybór osobników do reprodukcji może opierać się na selekcji turniejowej. Polega on na wyborze kilku osobników z populacji i wybrania z nich tego z najwyższym przystosowaniem. Dla każdego turnieju wybierać będziemy kilku losowych osobników z populacji, stawiać ich w szranki i wybierać zwycięzcę. Turnieje będą powtarzane tak, aby dzięki nim uzyskać wymaganą liczbę osobników na nową populację.

Algorytm selekcji ruletkowej

Ta metoda selekcji opiera się na prawdopodobieństwie, że osobnik zostanie wybrany do reprodukcji, przy czym szanse wyboru zależą od wartości funkcji przystosowania osobnika w stosunku do całej populacji. Im lepiej przystosowany osobnik, tym większe prawdopodobieństwo zostania wybranym. Tworzymy „koło ruletki" przydzielając każdemu osobnikowi zakres na kole proporcjonalny do prawdopodobieństwa jego wyboru. Losowo wybieramy punkty na kole ruletki. Znajdujemy osobnika, którego zakres na kole obejmuje wylosowany punkt. Taki osobnik będzie poddany reprodukcji. Powtarzając taką procedurę uzyskamy wymaganą liczbę osobników do utworzenia nowej populacji.

Algorytm selekcji progowej

W selekcji progowej wybiera się pewien procent najlepiej przystosowanych osobników z populacji, a resztę populacji losuje się z równomiernym prawdopodobieństwem. W ten sposób faworyzowane są osobniki o lepszym przystosowaniu, co sprzyja znajdowaniu lepszych rozwiązań. Badania, które

przeprowadzimy będą testowane dla różnych wartości procentu najlepszych osobników.

Dlaczego nie wykorzystujemy krzyżowania?

W kontekście rozważanego przez nas problemu rozmieszczania towarów w magazynie krzyżowanie nie jest pożądane, a nawet mogłoby prowadzić do pewnych problemów.

W naszym problemie konieczne jest przestrzeganie różnych ograniczeń, takich jak minimalna przestrzeń między towarami czy ograniczenia dotyczące kształtu magazynu, krzyżowanie może generować potomstwo, które łamie te ograniczenia, prowadząc do niepraktycznych i niewykonalnych rozwiązań. Zdefiniowanie krzyżowania dla problemu rozmieszczenia towarów w magazynie może być wyzwaniem ze względu na specyficzne cechy tego problemu. Potomstwo powstałe z osobników, które są już skuteczne w kontekście naszego problemu mogłoby być nieskuteczne. Dlatego nie stosujemy krzyżowania.

Funkcja celu

Postanowiliśmy, że funkcją celu będzie funkcja minimalizująca wartość rozwiązania. Jeśli osobnik (rozwiązanie) spełnia wszystkie założone kryteria dopuszczalności, opisane w sekcji *Krótki opis naszego rozwiązania*, to będzie miało ono wartość 0. Jeśli jest niedopuszczalne, to wartość tego rozwiązania będzie $+\infty$ lub jakaś pozytywna wartość, która zostanie obliczona na podstawie funkcji kary, opis której pokazano niżej. Zamierzamy zrobić badania uwzględniające te dwa przypadki, żeby zrozumieć które podejście jest lepsze (na razie wierzymy, że podejście z funkcją kary będzie lepsze).

Funkcja kary

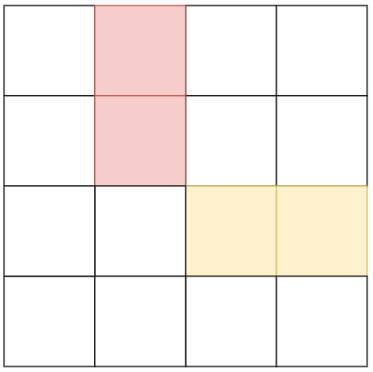
Na tym etapie projektu nie wiemy jakie ostatecznie wartości będą dla poszczególnych łamań zasad, jednak myślimy, że wstępne wartości przyjmiemy następujące:

1. Jeśli nie ma dostępnej klatki z wejściem, to do wartości rozwiązania doda się wartość 800.
2. Jeśli n klatek zostały zablokowane, to wartość rozwiązania wzrośnie na n*100.

3. Jeśli 1 klatka jest wspólna dla k towarów (na rysunku 4. pokazano przykład dla dwóch towarów), to wartość wzrośnie na k*200.

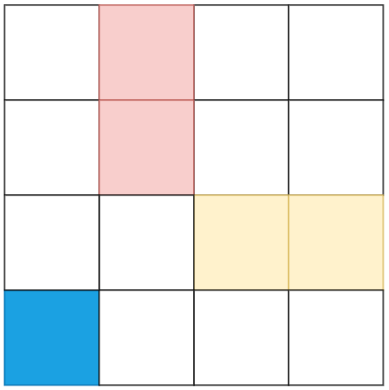
Algorytm znalezienia liczby zablokowanych klatek

Rozpatrzymy przypadek układu towarów pokazany niżej.



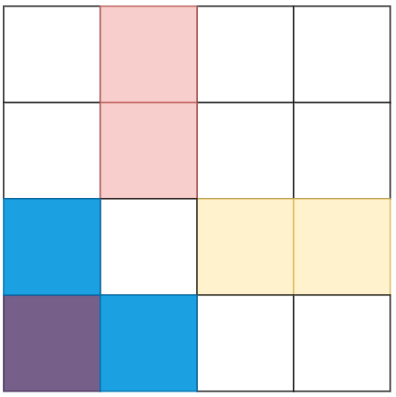
Rysunek 6. Przykładowe rozwiązanie

czerwonymi i żółtymi kolorami pokazano klatki z towarami, białe klatki to będą klatki z “niewiadomym” stanem, niebieskie oraz fioletowe będą klatki, do których magazynier ma dostęp. Niebieskie klatki będą w kolejce nieodwiedzonych klatek, a fioletowe to klatki odwiedzone. Najpierw policzymy liczbę klatek z “niewiadomym” stanem i zapamiętamy tą wartość. Załóżmy, że wejście do magazynu znajduje się w lewym dolnym rogu, więc oznaczymy ją kolorem niebieskim i dodamy ją do kolejki nieodwiedzonych klatek.



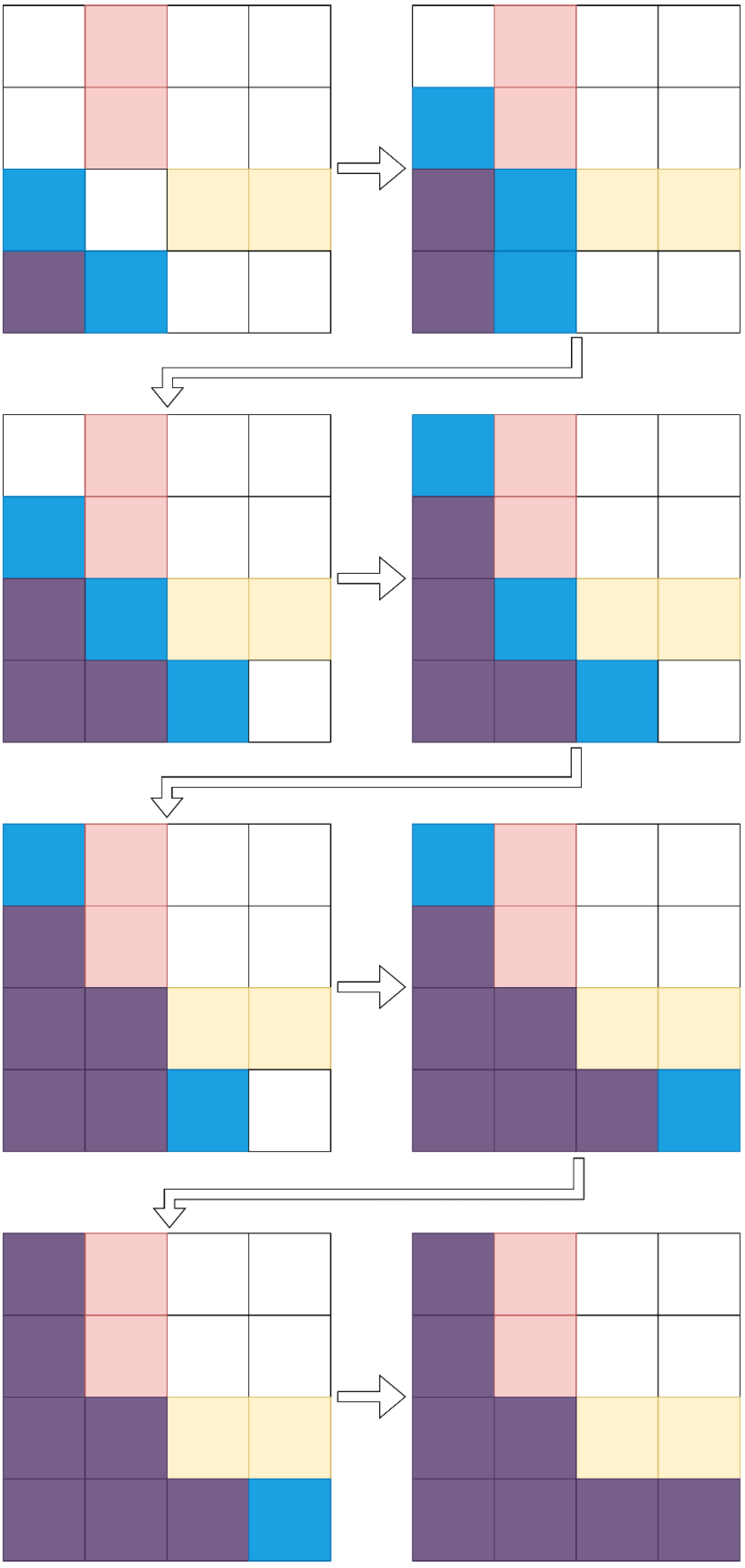
Rysunek 7. Początkowy stan rozwiązania

Następne działanie robimy w pętli (lub rekursywnie): bierzemy pierwszy w kolejce nieodwiedzonych klatek element i rozpatrujemy wszystkich sąsiadów znajdujących się na jedną klatkę wyżej, niżej, po lewej, po prawej stronach, które są klatkami z niewiadomym stanem i dodajemy je do kolejki. Potem pierwszą w kolejce nieodwiedzonych klatek dodamy do listy odwiedzonych klatek, więc będzie ona miała kolor fioletowy i usuniemy ją z kolejki nieodwiedzonych klatek. Aktualny stan klatek pokazano na rysunku niżej.



Rysunek 8. stan rozwiązania po jednej iteracji

Kolejne działania algorytmu pokazano na rysunkach poniżej, ale nie zostaną skomentowane, bo robimy te same działania, które zostały opisane wyżej. Przejście od stanu do stanu pokazano za pomocą strzałek. Dopóki kolejka nieodwiedzonych klatek nie będzie pusta, dopóty wykonujemy te same działania w pętli.



Rysunek 9. stany rozwiązań po kolejnych iteracjach

Jak można zauważyć, za pomocą algorytmu przeszukiwanie wszerz dowiedzieliśmy się do ilu klatek mamy dostęp. Jeśli ostatecznie wybierzemy wersję algorytmu, który uwzględnia tylko wejście do magazynu, które podał użytkownik, to w takim razie możemy z pewnością powiedzieć do ilu klatek nie mamy dostępu (w tym przypadku do 4). Natomiast, jeżeli wybierzemy wariant, że dopuszczalne rozwiązanie jest wtedy, gdy dostępna co najmniej jedna klatka "brzegowa", to wykonamy algorytm dla wszystkich klatek brzegowych, a następnie spośród wszystkich wyników wybierzemy minimalną wartość klatek, do której nie mamy dostępu (dla przypadku pokazanego wyżej to nadal będzie wartość 4, bo jeśli wybierzemy początek, na przykład, w prawym górnym rogu, to liczba niedostępnych klatek wyniesie 8, co jest większe od 4).

Opis aplikacji UI

Aplikacja będzie zawierała interfejs graficzny reprezentujący zadanie. Stworzymy prostą aplikację webową. Zakładamy, że powstanie okno wyboru zarówno dla definicji rozmiaru magazynu, jak i dla definiowania towarów. Przygotowane przez użytkownika dane trafią do naszego algorytmu. Ostatecznie po dokonaniu wszelkich obliczeń użytkownik zobaczy przygotowane rozwiązanie zawierające optymalne rozmieszczenie towarów, które poprzednio sam wprowadził oraz (jeśli wystąpi taki przypadek) to, ile towarów nie udało się przydzielić do magazynu o takich rozmiarach.

Opis badań (plan eksperymentów)

Będziemy badać wpływ różnych hiperparametrów: rozmiaru magazynu, rozmiaru i liczby towarów, rozmieszczenia wejścia (jeśli wybierzemy wersję z wejściem, które poda użytkownik), siły mutacji, prawdopodobieństwa mutacji. Zrobimy również badania dla różnych wariantów selekcji. Dla każdego zestawu hiperparametrów przeprowadzimy kilka (myślimy, że wartość 5 jest sensowną liczbą) eksperymentów i wybierzemy średnią wartość jako wartość oznaczającą "jakość" wybranych hiperparametrów. Wyniki będziemy reprezentować za pomocą wykresów oraz tabel. Przykładową tabelę pokazano niżej. Dane umieszczone tam nie są z rzeczywistych eksperymentów, jednak będzie podobna struktura tabel.

rozmiar magazynu	liczba towarów	rozmiary towarów	wejście	siła mutacji	prawdopodo-bieństwo mutacji	wariant selekcji	liczba iteracji	rozmieszczenie towarów	ocena
4,4	2	[[1,2], [2,1]]	0,0	1	0.2	turniejowa	2	[[1,1,2,3],[2,3,0,0]]	0
4,4	2	[[1,2], [2,1]]	0,0	1	0.4	turniejowa	1	[[1,1,2,3],[2,3,0,0]]	0

Tabela 1. Przykładowe obliczenias