



Algorytmy zaawansowane

Dokumentacja wstępna

Dawid Maksymowski

Artur Michalski

4 maja 2023

Streszczenie

Niniejszy dokument stanowi dokumentację wstępną algorytmu rozwiązującego problem Studni w czasie wielomianowym. Instancję problemu stanowi nieskierowany graf G o $n(k+1)$ wierzchołkach. W grafie wyróżnione jest n wierzchołków, które uznawane są za studnie, a pozostałe kn wierzchołków uznawane są za domy. Każda studnia jest w stanie dostarczyć wodę do dokładnie k domów, a koszt dostarczenia wody związany jest z odległością euklidesową pomiędzy studnią, a domem. Rozwiązanie problemu stanowi znalezienie przypisania domów do studni o minimalnym sumarycznym koszcie zgodnie z warunkami zadania. Zaproponowany w tym celu algorytm wykorzystuje metodę węgierską, a jego całkowita złożoność wynosi $O((kn)^3)$. W celu zweryfikowania poprawności działania algorytmu zostanie napisany program, którego wejście stanowić będzie plik zawierający informacje o liczbach k , n oraz lokalizacji studni oraz domów. Wyjście programu stanowić będzie plik zawierający informacje o optymalnym przypisaniu domów do studni oraz łącznym sumarycznym koszcie doprowadzenia wody do domów.

Spis treści

1	Opis problemu	3
2	Wprowadzenie teoretyczne	3
2.1	Twierdzenie Kőniga-Egerváry’ego	4
3	Opis rozwiązania	5
3.1	Opis algorytmu	5
3.2	Pseudokod	6
3.2.1	Preprocessing	6
3.2.2	Pętla główna	6
3.2.3	Powiększanie skojarzenia	7
3.2.4	Powiększenie potencjału	7
3.2.5	Algorytm węgierski	8
4	Analiza poprawności	9
4.1	Algorytm w każdej iteracji przybliża się do rozwiązania optymalnego	9
4.2	Aktualizacja funkcji potencjału nie modyfikuje wyznaczonego skojarzenia M	10
4.3	Funkcja p pozostaje poprawną funkcją potencjału dla rozpatrywanego grafu	10
5	Analiza złożoności obliczeniowej	11
5.1	Wyznaczenie wag krawędzi	11
5.2	Dodanie wierzchołków reprezentujących studnie w celu wyrównania liczności klas dwudzielności	11
5.3	Wyznaczanie ścieżki powiększającej	11
5.4	Aktualizacja potencjału wierzchołków	11
5.5	Algorytm węgierski	12
5.5.1	Wstępna analiza	12
5.5.2	Budowa algorytmu $O(n^3)$	12
5.5.3	Pseudokod algorytmu $O(n^3)$	12
6	Opis wejścia, wyjścia	13
6.1	Wejście	13
6.2	Wyjście	14

1 Opis problemu

Studnie Na płaszczyźnie znajduje się n studni oraz kn domów. Każda studnia zapewnia wodę k domom, a każdy dom może pobierać wodę tylko bezpośrednio ze studni (nie dopuszczamy pobierania wody za pośrednictwem innych domów). Należy znaleźć najtańsze sumaryczne doprowadzenie wody do domów, gdzie koszt doprowadzenia wody do domu d_i ze studni s_j jest równy odległości euklidesowej między punktami d_i i s_j .

Zauważmy, że w rozpatrywanym grafie G występują wyłącznie krawędzie łączące studnie z domami. Nie występują krawędzie łączące domy pomiędzy pozostałymi domami, czy studnie pomiędzy pozostałymi studniami. W związku z tym graf G jest grafem dwudzielnym, a klasy dwudzielności stanowią odpowiednio zbiory zawierające wyłącznie wierzchołki reprezentujące studnie oraz wierzchołki reprezentujące domy. Zauważmy również, że problem polega na znalezieniu optymalnego pod względem sumarycznego kosztu przypisania każdego z domów do dokładnie jednej studni z zachowaniem ograniczenia dotyczącego liczby domów do których woda może zostać doprowadzona za pomocą jednej studni. Pojedyncze przypisanie domu do studni można reprezentować za pomocą krawędzi łączącej wybrany wierzchołek ze zbioru domów z wybranym wierzchołkiem ze zbioru studni. Przypisanie wszystkich domów do studni można reprezentować za pomocą grafu, w którym stopień każdego wierzchołka reprezentującego studnię wynosi k , a każdy wierzchołek reprezentujący dom jest incydentny z dokładnie jedną krawędzią. Rozpatrzmy teraz operację polegającą na dodaniu do zbioru wierzchołków $(k-1)$ kopii każdego wierzchołka reprezentującego studnię. Zauważmy, że w tak zmodyfikowanym grafie klasy dwudzielności są równoliczne (jednemu domowi odpowiada jedna studnia). Rozpatrując teraz przypisanie wszystkich domów do studni uzyskujemy graf, którego zbiór krawędzi stanowi skojarzenie doskonałe (żadne dwie krawędzie nie są incydentne z tym samym wierzchołkiem oraz każdy wierzchołek należy do pewnej krawędzi). Tym samym wejściowy problem jest równoważny problemowi znajdowania skojarzenia doskonałego o najmniejszej wadze w zmodyfikowanym grafie dwudzielnym.

2 Wprowadzenie teoretyczne

Przypomnijmy kilka definicji teorii grafów:

Definicja (Graf dwudzielnny). *Graf dwudzielnny* to graf $G = (V, E)$, w którym zbiór wierzchołków V da się podzielić na dwa rozłączne podzbiory V_1 oraz V_2 tak, by żadne dwa wierzchołki w obrębie tego samego podzbioru V_i nie były połączone krawędzią.

Definicja (Pokrycie wierzchołkowe). *Pokrycie wierzchołkowe* grafu G to taki podzbiór jego wierzchołków, że każda krawędź z G jest incydentna do przynajmniej jednego wierzchołka z tego podzbioru.

Definicja (Skojarzenie). *Skojarzenie* w grafie $G = (V, E)$ to podzbiór krawędzi $M \subseteq E(G)$, w którym żadne dwie $v_1v_2, u_1u_2 \in M$ nie są incydentne z tym samym wierzchołkiem.

Definicja (Skojarzenie maksymalne). *Skojarzeniem maksymalnym* nazywamy takie skojarzenie w grafie G , że po dodaniu dowolnej krawędzi spośród krawędzi G do tego skojarzenia, przestaje ono być skojarzeniem.

Definicja (Skojarzenie największe). *Skojarzeniem największym* nazywamy takie skojarzenie w grafie G , że nie istnieje skojarzenie o większej liczbie krawędzi.

Definicja (Skojarzenie doskonałe). *Skojarzeniem doskonałym* nazywamy skojarzenie do którego należą wszystkie wierzchołki grafu G . Skojarzenie doskonałe jest również skojarzeniem maksymalnym oraz największym.

Definicja (Ścieżka naprzemienna). *Ścieżką naprzemienną* w grafie G o skojarzeniu M nazywamy ścieżkę v_1, v_2, \dots, v_n jeśli co druga krawędź ze ścieżki należy do skojarzenia.

Definicja (Ścieżka powiększająca). *Ścieżką naprzemienną* nazywamy *ścieżką powiększającą*, jeśli pierwszy i ostatni wierzchołek ścieżki są nieskojarzone.

Znane jest również twierdzenie Kóniga dla grafów nieważonych:

Twierdzenie 1 (Kóniga). *Liczba krawędzi największego skojarzenia w grafie dwudzielnym jest równa liczbie wierzchołków najmniejszego pokrycia wierzchołkowego.*

Twierdzenie to jest jednym z wielu twierdzeń MIN-MAX zachodzących w teorii grafów, które wyrażają dualność między konstrukcjami wierzchołkowymi oraz krawędziowymi w grafach. W naszym przypadku jednak przydatne będzie twierdzenie ogólniejsze.

2.1 Twierdzenie Kóniga-Egerváry’ego

Okazuje się, że twierdzenie 1 można również uogólnić dla grafów ważonych. Może ono przyjąć różne sformułowania, natomiast w tym dokumencie zostanie opisana wersja przydatna w konstrukcji algorytmu znajdującego skojarzenia o najmniejszej sumie wag. W tym celu wprowadzamy kilka kolejnych definicji.

W dalszej części dokumentu przyjmujemy następujące oznaczenia:

$G = (V, E)$ - graf dwudzielnym, ważony z wagami na krawędziach,

$V = W \cup H$ - podział wierzchołków na dwie równoliczne klasy dwudzielności,

$w : E \rightarrow \mathbb{R}$ - funkcja wagowa krawędzi.

Definicja (Potencjał wierzchołka). ***Funkcją potencjału** nazwiemy każdą funkcję $p : V \rightarrow \mathbb{R}$ spełniającą warunek (zwany właściwością funkcji potencjału):*

$$p(v_1) + p(v_2) \leq w(v_1v_2), \forall v_1v_2 \in E$$

Wówczas **potencjałem wierzchołka** v nazwiemy wartość funkcji potencjału $p(v)$.

Definicja (Wartość potencjału). ***Wartością potencjału** p nazwiemy wartość:*

$$\sum_{v \in V} p(v)$$

Definicja (Krawędzie ciasne). *Krawędź $v_1v_2 \in E$ nazwiemy **krawędzią ciasną**, jeżeli*

$$p(v_1) + p(v_2) = w(v_1v_2)$$

Definicja (Krawędzie luźne). *Krawędź $v_1v_2 \in E$ nazwiemy **krawędzią luźną**, jeżeli nie jest krawędzią ciasną.*

Definicja (Zacieśnienie krawędzi). ***Zacieśnieniem** krawędzi luźnej uv nazwiemy taką modyfikację funkcji potencjału, że uv staje się krawędzią ciasną oraz zachowana jest właściwość funkcji potencjału.*

Twierdzenie 2 (Kóniga-Egerváry’ego). *Jeżeli G - graf ważony, z wagami na krawędziach, to:*

$$\text{waga skojarzenia doskonałego o najmniejszej wadze} = \text{wartość największego potencjału}$$

Niezwykle przydatne jest również twierdzenie 2, zachodzące dla grafów dwudzielnych, które w szczególny sposób wyraża intuicję stojącą za opisywanym w dalszej części dokumentu algorytmem:

Twierdzenie 3 (Kuhna-Munkresa). *Niech:*

$G = (V, E)$ - graf ważony, z wagami na krawędziach,

$E_p = \{(x, y) \in E : p(x) + p(y) = w(x, y)\}$ - zbiór krawędzi ciasnych w G względem potencjału p ,

$G_p = (V, E_p)$ - graf krawędzi ciasnych grafu G .

*Funkcja p jest poprawną funkcją potencjału dla wszystkich krawędzi z M
oraz M jest skojarzeniem doskonałym w grafie G_p*

$$\Longleftrightarrow$$

M jest skojarzeniem doskonałym o najmniejszej wadze w grafie G

Intuicja stojąca za twierdzeniami 2 i 3 jest następująca: waga dowolnego skojarzenia doskonałego wynosi co najmniej tyle, ile wartość dowolnego (tzn. każdego) potencjału w grafie (co wynika bezpośrednio z definicji funkcji potencjału oraz z faktu, że skojarzenie doskonałe pokrywa *wszystkie* wierzchołki grafu - każdy wierzchołek dokładnie raz). Ponadto znalezienie skojarzenia doskonałego o mniejszej wadze jest równoważne znalezieniu potencjału o większej wartości. Tym samym, dla każdego skojarzenia istnieje potencjał, który jest mu odpowiadający (jest *dualny* do niego). Możemy się więc w ten sposób iterować: waga skojarzenia oraz wartość potencjału muszą „spotkać się” w pewnym miejscu, o czym mówi tw. 2. Ważny wniosek z tego wynikający jest taki, że *problem poszukiwania skojarzenia doskonałego o najmniejszej sumie wag* możemy zamienić na *problem poszukiwania potencjału o największej wartości*. Gdy już znajdziemy taki potencjał, możemy skonstruować na podstawie niego skojarzenie, korzystając z cechy *dualności*. Okazuje się, że cechą tą jest właściwość **ciasności krawędzi**:

skojarzenie doskonałe o najmniejszej wadze składa się tylko z krawędzi ciasnych znalezionej potencjału.

Lemat 4 (Berge’a). *Skojarzenie M w grafie G jest największe wtedy i tylko wtedy, gdy nie istnieje w G ścieżka powiększająca względem M .*

Dowodzenie przedstawionych w tym rozdziale twierdzeń wychodzi poza zakres dokumentu.

3 Opis rozwiązania

Oznaczmy przez n licznosc zbioru zawierającego wyłącznie wierzchołki reprezentujące studnie oraz przez k liczbę domów, które mogą zostać zaopatrzone w wodę przez jedną studnię. Zauważmy, że zbiór wierzchołków zawierający wyłącznie wierzchołki reprezentujące studnie (dalej oznaczany jako W) oraz zbiór wierzchołków zawierający wyłącznie wierzchołki reprezentujące domy (dalej oznaczany jako H) stanowią klasy dwudzielności. Wynika, to z faktu, że w rozpatrywanym grafie istnieją wyłącznie krawędzie łączące domy ze studniami. Nie występują krawędzie pomiędzy dwoma wierzchołkami należącymi do zbioru studni, ani krawędzie pomiędzy dwoma wierzchołkami należącymi do zbioru domów. W przypadku gdy każda studnia dostarcza wodę więcej niż jednemu domowi klasy dwudzielności te nie są równoliczne. Rozszerzmy, więc zbiór W poprzez dodanie $k - 1$ instancji każdego wierzchołka należącego do W . Tym samym otrzymujemy równoliczne klasy dwudzielności, a w zbiorze W występuje dokładnie k instancji każdej ze studni. Rozpatrzmy teraz dowolne przypisanie domów do studni zgodne z warunkami zadania. Dla rozpatrywanych zbiorów W oraz H przypisanie to odpowiada połączeniu każdego wierzchołka należącego do zbioru H z dokładnie jednym wierzchołkiem należącym do zbioru W . Zauważmy, że ze względu na dwudzielność zbiorów W oraz H przypisanie domów do studni odpowiada znalezieniu skojarzenia doskonałego w zmodyfikowanym grafie. Szukane przypisanie o sumarycznym minimalnym koszcie odpowiada więc skojarzeniu doskonałemu o minimalnym koszcie. W celu znalezienia rozwiązania zadania, można więc skorzystać z algorytmów znajdujących skojarzenie doskonałe o minimalnym koszcie w grafie dwudzielnym. Jednym z takich algorytmów jest Algorytm Węgierski.

3.1 Opis algorytmu

Algorytm w całości opiera swoje działanie o twierdzenie 2 i dla jego pełnego zrozumienia bardzo przydatne może być zrozumienie nie tylko treści, lecz również intuicji stojącej za tym twierdzeniem.

Główna pętla algorytmu **Hungarian** została opisana w formie pseudokodu w podsekcji 3.2. Najważniejszymi strukturami, na których działa algorytm są funkcja potencjału p oraz podgraf krawędzi ciasnych G_p . Skojarzenie M jest rozbudowywane w trakcie działania algorytmu na podstawie wartości w p i G_p .

Ideę działania algorytmu węgierskiego można zamknąć w trzech skondensowanych zdaniach. W ogólnym jego kroku znajdujemy największe skojarzenie w G_p , korzystając z algorytmu ścieżek powiększających i znalezione skojarzenie przypisujemy do M . Na tym etapie nie bierzemy pod uwagę wag krawędzi. Jeżeli w G_p nie ma już żadnej ścieżki powiększającej, ale wciąż istnieje wierzchołek,

który nie jest pokryty przez M , szukamy potencjału o większej wartości takiego, który nie zmodyfikuje skojarzenia M (tzn. wszystkie krawędzie tego skojarzenia będą wciąż należeć do nowo-obliczonego G_p), ale umożliwi odnalezienie ścieżki powiększającej w kolejnym kroku. Robimy to tak długo, jak skojarzenie nie jest doskonałe.

3.2 Pseudokod

Ta część dokumentu przedstawia pseudokod proponowanego algorytmu.

3.2.1 Preprocessing

W ramach preprocessingu wykonywane są trzy procedury pomocnicze. Ich wykonanie jest niezbędne do poprawnej interpretacji danych wejściowych, w związku z czym ich złożoność czasowa została zanalizowana w sekcji 5. Nie są one jednak kluczowe z punktu widzenia projektowania algorytmu. Te operacje to:

- utworzenie grafu G w oparciu o dane zawarte w pliku podanym na wejściu programu,
- wyznaczenie wag krawędzi,
- rozszerzeniu zbioru stanowiącego wierzchołki reprezentujące studnie poprzez dodanie $(k-1)$ kopii każdej studni.

Tak utworzony i przetworzony graf G stanowi następnie wejście algorytmu węgierskiego (alg. 5), który wyznacza optymalne przypisanie domów do studni znajdując skojarzenie doskonałe o minimalnym koszcie.

Algorithm 1 ExtendWells

Input: $G = (V, E)$

$V = W \cup H$ - podział wierzchołków na klasy dwudzielności

Output: G - zmodyfikowany graf G poprzez powielenie każdej studni $(k-1)$ razy

```

1: for all  $v$  in  $W$  do
2:   dodaj  $(k-1)$  kopii  $v$  do  $W$ 
3: end for
4: return  $G$ 
```

Algorithm 2 CalculateDistances

Input: Loc - tablica zawierająca lokalizacje wierzchołków ($Loc[i] - (x_i, y_i)$)

Output: $Dist$ - macierz kosztów doprowadzenia wody przez studnię. ($Dist[i, j]$ - koszt doprowadzenia wody za pomocą i -tej studni do j -tego domu)

```

1:  $Dist \leftarrow$  pusta tablica
2: for all  $w$  in  $W$  do
3:   for all  $h$  in  $H$  do  $Dist[w, h] \leftarrow distance(Loc[w], Loc[h])$ 
4:   end for
5: end for
6: return  $Dist$ 
```

3.2.2 Pętla główna

Bazując na twierdzeniu 3, główna pętla algorytmu mogłaby wyglądać następująco:

Dopóki skojarzenie M nie jest doskonałe:

1. Jeśli istnieje ścieżka powiększająca względem M w grafie G_p : powiększ M
2. Wpp. - znajdź lepszą funkcję p' tż. $G_{p'} \subset G_p$

Warto zauważyć, że każda z iteracji pętli albo zwiększy rozmiar skojarzenia M albo grafu G_p , więc pętla zakończy swoje działanie w skończonym czasie.

Co więcej, kiedy ten proces się zakończy, M będzie skojarzeniem doskonałym w grafie G_p dla p – funkcji spełniającej warunek potencjału. Oznacza to, że M będzie skojarzeniem doskonałym o najmniejszej wadze na mocy twierdzenia 3.

Dla pełnego zrealizowania algorytmu, należy się zastanowić, jak najlepiej wykonać oba z wyżej wymienionych kroków, co opisuje dalsza część dokumentu.

3.2.3 Powiększanie skojarzenia

Ścieżka powiększająca względem dowolnego skojarzenia zaczyna się i kończy wierzchołkami, które nie są incydentne z żadną krawędzią należącą do skojarzenia. Skojarzenie to możemy wówczas w prosty sposób powiększyć, odwracając przynależność do skojarzenia każdej krawędzi ze ścieżki powiększającej (mówimy wówczas, że *powiększamy skojarzenie względem ścieżki powiększającej*). Jest to więc nieskomplikowana procedura, którą możemy zrealizować, posilkując się algorytmem BFS.

Procedurę znajdowania największego skojarzenia prezentuje algorytm 3. Jako argumenty przyjmuje podgraf G_p krawędzi ciasnych, skojarzenie M oraz sam początkowy graf G (wraz z podziałem na klasy dwudzielności).

W **linii 1** jest tworzony graf, który jest skierowaną kopią grafu G_p zgodnie z zasadą: jeśli dana krawędź hw należy do M , krawędź jest skierowana w kierunku wierzchołka $h \in H$. Wpp. jest skierowana w kierunku wierzchołka $w \in W$.

W **liniach 2-3** tworzymy zbiór takich wierzchołków z obu zbiorów dwudzielności, które nie są pokryte przez dotychczas zbudowane M .

Następnie, w **linii 4**, szukamy ścieżki powiększającej w grafie \vec{G}_p , który stworzyliśmy w **linii 1**. Jeżeli taka istnieje – oznacza to, że możemy „odwrócić” przynależność jej krawędzi do skojarzenia. Innymi słowy, rozmiar skojarzenia zwiększył się o 1.

Linie **1-3** zostały opisane dla uproszczenia oraz czytelności, lecz można je w praktyce pominąć w implementacji, korzystając z innych struktur (np. zwykłej tablicy), które będą przechowywały informację, czy dany wierzchołek należy do zbioru W czy H . Podobnie graf \vec{G}_p : jego główną funkcją jest zapewnienie, że znaleziona ścieżka jest rzeczywiście *naprzemienna*. Możliwa jest drobna modyfikacja algorytmu, tak, by naprzemiennosc była sprawdzana w trakcie budowania ścieżki, bez tworzenia nowego grafu. Operacje te można więc pominąć przy szacowaniu złożoności obliczeniowej.

Algorithm 3 AugmentingPath

Input: $G = (W \cup H; E)$

M - skojarzenie

G_p - podgraf krawędzi ciasnych grafu G

Output: M zmodyfikowane - największe skojarzenie w G_p

- 1: $\vec{G}_p \leftarrow$ graf skierowany na bazie M i p
 - 2: $R_H \leftarrow$ zb. wierzch. z H niepokrytych przez M
 - 3: $R_W \leftarrow$ zb. wierzch. z W niepokrytych przez M
 - 4: $path \leftarrow$ ścieżka powiększająca z R_H do R_W w grafie \vec{G}_p
 - 5: $M' \leftarrow$ skojarzenie M powiększone względem $path$
 - 6: **return** (M', Z)
-

3.2.4 Powiększenie potencjału

Drugą część pętli stanowi powiększenie potencjału p w grafie. Aby procedura powiększenia miała sens, chcielibyśmy, aby $G_{p'} \subset G_p$, tzn. algorytm poszerzał podgraf krawędzi ciasnych o nowe krawędzie, ale zostawiał poprzednie. Dowód na to, że własność tę spełnia algorytm 4 został przedstawiony w sekcji 4.2.

Linie 1-3 oznaczają te same struktury co w algorytmie 3. Zbiór Z jest natomiast tworzony wyłącznie do wyznaczenia wartości Δ . Jest to najmniejsza możliwa wartość, o którą można powiększyć

potencjał pewnego wierzchołka, która jednocześnie zacieśni co najmniej jedną krawędź¹. Dzięki temu w następnym kroku pętli graf \vec{G}_p będzie posiadał ścieżkę powiększającą.

Poprawność dalszej, tj. właściwej części algorytmu 4 została dokładnie opisana w podsekcji 4.2.

Algorithm 4 ImprovePotential

Input: $G = (W \cup H; E)$

M - skojarzenie

Output: p - zaktualizowana funkcja potencjału

```

1:  $\vec{G}_p \leftarrow$  graf skierowany na bazie  $M$  i  $p$ 
2:  $R_H \leftarrow$  zb. wierzch. z  $H$  niepokrytych przez  $M$ 
3:  $R_W \leftarrow$  zb. wierzch. z  $W$  niepokrytych przez  $M$ 
4:  $Z \leftarrow$  zb. wierzch. osiągalnych z  $R_H$  w grafie  $\vec{G}_p$ 
5:  $\Delta \leftarrow \min \{w(i, j) - p(i) - p(j) : i \in Z \cap H, j \in W - Z\}$ 
6: for all  $v$  in  $Z \cap H$  do  $p(v) \leftarrow p(v) + \Delta$ 
7: end for
8: for all  $v$  in  $Z \cap W$  do  $p(v) \leftarrow p(v) - \Delta$ 
9: end for
10: return  $p$ 

```

3.2.5 Algorytm węgierski

Algorytm 5 stanowi pseudokod stworzony na podstawie kroków opisanych w 3.2.2 oraz algorytmów 3, 4. Można zauważyć, że niektóre z instrukcji (**linie 5-7** w alg. 5) tych pseudokodów się powtarzają, w związku z czym zostały one wyniesione bezpośrednio na poziom głównej pętli.

Linie 1-3 stanowią inicjalizację. W pierwszym kroku ustawiamy startowe wartości funkcji potencjału. Najprostszą, poprawną metodą na ustalenie tych wartości jest przypisanie każdemu wierzchołkowi wartości 0. Proponujemy jednak ustalenie potencjału na 0 dla wszystkich domów, a dla dowolnej studni s – wzięcie $\min_{sv \in E(G)} c(sv)$, tj. najmniejszej wagi spośród wszystkich wag krawędzi incydentnych z s . Znalezienie tych wag nie ma wpływu na sumaryczną złożoność (sekcja 5), a może zmniejszyć liczbę wykonań głównej pętli o zauważalną wartość.

Na podstawie znalezionej funkcji potencjału budujemy podgraf grafu G składający się z wszystkich jego wierzchołków, lecz tylko tych krawędzi, które są ciasne. Graf ten zazwyczaj nie będzie posiadał żadnych krawędzi, poza przypadkami, w których jakaś studnia znajduje się bezpośrednio w domu (odległość 0).

Dalszą część stanowią połączone algorytmy 3 i 4.

¹To w tym miejscu dzieje się cała "magia" algorytmu węgierskiego.

Algorithm 5 Hungarian

Input: $G = (V, E)$ $V = W \cup H$ - podział wierzchołków na klasy dwudzielności w - funkcja wag krawędzi**Output:** M - skojarzenie doskonałe o minimalnym koszcie zawierające optymalne przypisanie domów do studni

```
1:  $p[] \leftarrow$  startowa funkcja potencjału
2:  $G_p \leftarrow$  podgraf krawędzi ciasnych grafu  $G$ 
3:  $M \leftarrow \emptyset$ 
4: while  $M$  nie jest doskonałe do
5:    $\vec{G}_p \leftarrow$  graf skierowany na bazie  $M$  i  $p$ 
6:    $R_H \leftarrow$  zb. wierzch. z  $H$  niepokrytych przez  $M$ 
7:    $R_W \leftarrow$  zb. wierzch. z  $W$  niepokrytych przez  $M$ 
8:    $path \leftarrow$  ścieżka powiększająca z  $R_H$  do  $R_W$  w grafie  $\vec{G}_p$ 
9:   if  $path \neq \emptyset$  then
10:     $M \leftarrow$  skojarzenie  $M$  powiększone względem  $path$  ▷ AugmentingPath
11:   else
12:     $Z \leftarrow$  zb. wierzch. osiągalnych z  $R_H$  w grafie  $\vec{G}_p$  ▷ ImprovePotential
13:     $\Delta \leftarrow \min \{w(i, j) - p(i) - p(j) : i \in Z \cap H, j \in W - Z\}$ 
14:    for all  $v$  in  $Z \cap H$  do  $p(v) \leftarrow p(v) + \Delta$ 
15:    end for
16:    for all  $v$  in  $Z \cap W$  do  $p(v) \leftarrow p(v) - \Delta$ 
17:    end for
18:  end if
19: end while
20: return  $M$ 
```

4 Analiza poprawności

Operacje mające na celu przygotowanie wejściowej instancji problemu takie jak rozszerzenie zbioru studni tak, aby klasy dwudzielności były równoliczne, czy wyznaczenie wag krawędzi w oparciu o współrzędne wierzchołków należą do operacji dla których dowód poprawności jest trywialny. Z tego powodu przedmiotem dalszych rozważań będzie poprawność działania algorytmu węgierskiego. W wykazaniu poprawności algorytmu posłużymy się przytoczonym twierdzeniem 2. Na jego mocy suma wag krawędzi skojarzenia doskonałego o najmniejszej wadze odpowiada sumie potencjałów wierzchołków należących do pokrycia wierzchołkowego o największej wartości potencjału. W związku z tym wykazanie, że opisany algorytm znajduje pokrycie wierzchołkowe o największej wartości potencjału, będzie dowodziło również poprawności algorytmu w wyznaczaniu skojarzenia doskonałego o minimalnym koszcie. W tym celu w kolejnych podsekcjach zostaną wykazane podstawowe własności algorytmu, które stanowią o jego poprawności:

- Algorytm w każdej iteracji przybliża się do rozwiązania optymalnego
- Aktualizacja funkcji potencjału nie modyfikuje wyznaczonego skojarzenia M
- Funkcja p pozostaje poprawną funkcją potencjału dla rozpatrywanego grafu

4.1 Algorytm w każdej iteracji przybliża się do rozwiązania optymalnego

Pokażemy, że tak długo jak uzyskane skojarzenie M nie jest największe, algorytm wykonuje krok przybliżający do rozwiązania optymalnego - poprzez rozszerzenie skojarzenia lub zacieśnienie co najmniej jednej krawędzi. Tym samym w każdym kroku zachodzi co najmniej jeden z poniższych warunków:

1. M jest skojarzeniem największym

2. G_p zawiera ścieżkę powiększającą
3. G zawiera ścieżkę zaczynającą się wierzchołkiem należącym do zbioru R_H i kończącą luźną krawędzią prowadzącą do wierzchołka należącego do zbioru $W - Z$

Jeżeli M jest skojarzeniem największym naturalnie spełniony jest pierwszy warunek. Przypuśćmy więc, że M nie jest skojarzeniem największym. Wtedy na mocy lematu Berge'a (4) w grafie G istnieje ścieżka powiększająca P względem M . Niekoniecznie jednak ścieżka musi istnieć w grafie G_p . Niemniej jednak z definicji skojarzenia M wynika, że każda krawędź należąca do $P \cap M$ jest ciasna. Skojarzenie M zostało bowiem wyznaczone w oparciu o znaną ścieżkę powiększającą w grafie G_p stanowiącym podgraf krawędzi ciasnych grafu G . Pozostałe krawędzie należące do P mogą być luźne i z tego powodu nie znajdują się w grafie G_p . Jeden z końców ścieżki P znajduje się w wierzchołku należącym do zbioru H (wierzchołek ten należy również do zbioru R_H , co wynika z faktu, że P jest ścieżką powiększającą wobec M), a drugi koniec znajduje się w wierzchołku należącym do zbioru W . Bez straty ogólności przyjmijmy, że ścieżka zaczyna się wierzchołkiem należącym do zbioru H . Jeżeli każda krawędź należąca do ścieżki P jest ciasna, wtedy P jest szukaną ścieżką powiększającą w grafie G_p i spełniony jest drugi warunek. W przeciwnym wypadku P zawiera luźną krawędź. Niech uv będzie pierwszą taką krawędzią. Jeżeli $v \notin Z$ wtedy podścieżka złożona z kolejnych krawędzi ścieżki P , aż do wierzchołka v jest szukaną ścieżką zakończoną luźną krawędzią i spełniony jest warunek trzeci. W przeciwnym przypadku wierzchołek v jest osiągalny z pewnej innej ścieżki Q złożonej z ciasnych krawędzi i zaczynającej się w wierzchołku należącym do zbioru R_H . Niech P_v będzie podścieżką P zaczynającą się w v i zawierającą kolejne krawędzie ze ścieżki P . Niech teraz P' będzie ścieżką utworzoną poprzez sumę ścieżek Q oraz P_v . Zauważmy, że ścieżka P' jest ścieżką powiększającą wobec M w G złożoną z o co najmniej jedną luźną krawędź mniej względem ścieżki P . Ścieżkę P można zastąpić ścieżką P' i przeprowadzone rozumowanie można wielokrotnie iterować (formalnie wykonując indukcję po liczbie luźnych krawędzi) do momentu aż nie zostanie znaleziona ścieżka powiększająca w G_p (spełnienie drugiego warunku) lub w G zostanie znaleziona ścieżka zakończona luźną krawędzią (spełnienie trzeciego warunku).

4.2 Aktualizacja funkcji potencjału nie modyfikuje wyznaczonego skojarzenia M

Aby wykazać, że aktualizacja funkcji potencjału nie wpływa na skojarzenie M wystarczy wykazać, że oba wierzchołki incydentne z dowolną krawędzią należącą do skojarzenia M jednocześnie należą lub nie należą do zbioru Z . Rozpatrzmy dowolną krawędź vu ze zbioru W do H należącą do skojarzenia M . Naturalnie jeżeli wierzchołek v należy do zbioru Z , wtedy wierzchołek u również należy do zbioru Z , co wynika z faktu, że skojarzenie M zawiera wyłącznie krawędzie ciasne. Załóżmy teraz nie wprost, że $u \in Z$, ale $v \notin Z$. u nie może należeć do zbioru R_H ponieważ należy do krawędzi zawartej w skojarzeniu, więc istnieje pewna skierowana ścieżka złożona z ciasnych krawędzi z wierzchołka zawartego w R_H do u . Ścieżka ta nie zawiera v , ponieważ z założenia nie należy on do zbioru Z . W związku z tym wierzchołek bezpośrednio poprzedzający u w rozpatrywanej ścieżce jest jakimś innym wierzchołkiem $v' \in W$. Wtedy $v'u$ jest ciasną krawędzią z W do H i należy do M . Ale wtedy M zawiera dwie krawędzie incydentne z wierzchołkiem u , co stanowi sprzeczność z M byciem skojarzeniem. Tym samym wierzchołki będące końcami każdej krawędzi należącej do zbioru M jednocześnie należą lub nie należą do zbioru Z .

4.3 Funkcja p pozostaje poprawną funkcją potencjału dla rozpatrywanego grafu

Aby wykazać, że funkcja p pozostaje poprawną funkcją potencjału dla rozpatrywanego grafu wystarczy wykazać, z definicji potencjału, że waga żadnej krawędzi jest nie mniejsza od sumy potencjałów incydentnych wierzchołków. Dla krawędzi należących do zbioru M zostało to już wykazane w ramach dowodu z poprzedniej podsekcji. Rozpatrzmy więc dowolną krawędź uv zaczynającą się w zbiorze H i kończącą w zbiorze W . Jeżeli $p(u)$ jest powiększane o wyznaczoną wartość Δ wtedy albo $v \in Z \cap W$ i wtedy $y(v)$ jest zmniejszane o wartość Δ , co nie powoduje zmiany sumarycznej wartości

potencjału dla rozpatrywanej krawędzi uv lub $v \in W \setminus Z$ i wtedy z definicji Δ zachodzi nierówność $y(u) + y(v) + \Delta \leq w(u, v)$. W związku z tym funkcja p pozostaje poprawną funkcją potencjału.

5 Analiza złożoności obliczeniowej

W celu wyznaczenia złożoności algorytmu została przeprowadzona analiza złożoności poszczególnych elementów algorytmu.

5.1 Wyznaczenie wag krawędzi

Koszt związany z połączeniem danej studni oraz danego domu odpowiada odległości euklidesowej pomiędzy rozpatrywaną studnią, a domem. Aby wyznaczyć koszt dla wszystkich krawędzi, należy rozpatrzyć wszystkie możliwe pary: studnia-dom. Liczność zbioru studni wynosi n , a liczność zbioru domów kn . Koszt wyznaczenia odległości dla pojedynczej pary jest stały. Tym samym złożoność czasowa wyznaczenia wag krawędzi wynosi $O(kn^2)$.

5.2 Dodanie wierzchołków reprezentujących studnie w celu wyrównania liczności klas dwudzielności

W przypadku gdy $k \neq 1$ klasy dwudzielności (zbiór wierzchołków reprezentujących studnie oraz zbiór wierzchołków reprezentujących domy) nie są równoliczne. W tym wypadku należy rozszerzyć zbiór studni poprzez dodanie $(k - 1)$ kopii każdego wierzchołka reprezentującego studnie wraz z krawędziami reprezentującymi połączenia z każdym domem. Operację tę należy wykonać dla n wierzchołków reprezentujących studnie, a w ramach pojedynczej operacji tworzone jest $(k - 1)$ kopii wierzchołka wraz z kn krawędziami. Tym samym złożoność czasowa wyrównywania liczności klas dwudzielności wynosi $O((kn)^2)$. W celu uproszczenia dalszych analiz wartość n będzie utożsamiana od tej pory z licznoscią zbioru wierzchołków otrzymanego grafu (dotychczas liczność zbioru wierzchołków wejściowego grafu była wyrażona jako $(k + 1)n$).

5.3 Wyznaczanie ścieżki powiększającej

Procedura przedstawiona w ramach algorytmu 3 pozwala na znajdowanie w sposób optymalny ścieżki powiększającej dla skojarzenia M w grafie złożonym wyłącznie z krawędzi ciasnych. Czas działania procedury może być ograniczony przez rozmiar problemu i okazuje się, że jest wielomianowy.

Skończoność procedury wynika z następujących faktów:

- Jeżeli skojarzenie nie jest największe, istnieje wierzchołek v z którym żadna z krawędzi należących do skojarzenia nie jest incydentna
- Na mocy lematu Berge'a 4 istnieje więc ścieżka powiększająca
- Z własności ścieżki powiększającej wynika, że rozmiar skojarzenia w ramach pojedynczej iteracji jest zwiększany o jeden

Złożoność pojedynczego wykonania procedury jest rzędu $O(|E|)$ co wynika z konieczności przejrzenia wszystkich krawędzi i sprawdzeniu czy należą one do skojarzenia M przy budowie ścieżki. Naturalnie liczność zbioru krawędzi E jest ograniczona z góry przez kwadrat liczby wierzchołków grafu, tym samym zachodzi również: $|E| \leq n^2$. Złożoność całej procedury wynosi więc $T(n) \in O(n^2)$.

5.4 Aktualizacja potencjału wierzchołków

Rozpatrzmy złożoność kolejnych operacji związanych z procedurą aktualizacji potencjału wierzchołków. W pierwszym kroku wyznaczana jest wartość Δ stanowiąca minimum z różnicy wagi krawędzi, a potencjałami incydentnych do niej wierzchołków. W tym kroku rozpatrywane są wyłącznie krawędzie zaczynające się w wierzchołkach będącymi domami i są osiągalne z R_H , a kończące się wierzchołkami będącymi studnią, które nie są osiągalne z R_H . Obliczanie wartości delta dla pojedynczej krawędzi

odbywa się w czasie stałym, a operacja ta wykonywana jest dla co najwyżej każdej krawędzi grafu. Tym samym złożoność wyznaczania wartości Δ wynosi $O(n^2)$.

W kolejnym kroku aktualizowany jest potencjał dla odpowiednio wszystkich wierzchołków należących do zbioru wierzchołków reprezentujących domy, które nie są pokryte przez skojarzenie M oraz wierzchołków należących do zbioru wierzchołków reprezentujących studnie, które nie są pokryte przez skojarzenie M . Aktualizacja potencjału dla pojedynczego wierzchołka odbywa się w czasie stałym, a liczba wykonywanych aktualizacji jest ograniczona z góry przez liczbę wszystkich wierzchołków w grafie. Stąd złożoność operacji aktualizacji potencjału wierzchołków wynosi $O(n)$.

Sumaryczna złożoność procedury aktualizacji potencjału wierzchołków wynosi więc $O(n^2)$.

5.5 Algorytm węgierski

5.5.1 Wstępna analiza

Przyjrzyjmy się pseudokodowi 5. Otóż wstępna analiza tego algorytmu może wskazywać na to, że jego złożoność obliczeniowa wynosi $O(n^4)$. Rozpatrzmy argumentację potwierdzającą ten fakt.

Operacja powiększania skojarzenia **AugmentingPath** wykona się co najwyżej $O(n)$ razy, z uwagi na ograniczoną licznosc największego skojarzenia. Na każdą z tych operacji przypada natomiast w pesymistycznym przypadku aż $O(n)$ operacji **ImprovePotential**, zanim w grafie \vec{G}_p powstanie jakaś kolejna ścieżka powiększająca. Widzimy więc, że pętla z linii **4-19** wykona się $O(n^2)$ razy. Jak dowiedliśmy wcześniej, złożoność linii **8** wynosi $O(n^2)$, linii **10** – $O(n)$, a linii **12-17** – $O(n^2)$. Stąd sumaryczna złożoność tej wersji algorytmu wynosi $O(n^4)$.

5.5.2 Budowa algorytmu $O(n^3)$

Opisana w tym dokumencie wersja algorytmu węgierskiego jest przydatna do łatwiejszego zrozumienia jego zasady działania oraz analizy poprawności. Jednak dla osiągnięcia optymalnej złożoności czasowej niezbędne jest wprowadzenie pewnych udoskonaleń. Spójrzmy więc ponownie na algorytm, tym razem bardziej krytycznym okiem.

Linia 8 Zwróćmy uwagę na to, że nie mamy żadnych szczególnych wymagań dot. tego, jak ma wyglądać ścieżka *path* – nie musi być najkrótsza, o najmniejszej wadze itp. Ścieżka *path* jest DOWOLNĄ ścieżką powiększającą w grafie \vec{G}_p . Dzięki tej obserwacji nie musimy przeglądać całego zbioru R_H . Możemy wybrać dowolny wierzchołek u ze zbioru R_H i od niego zacząć budowę ścieżki. Wiemy, że taka ścieżka musi istnieć w G , ponieważ skojarzenie końcowe, które szukamy pokrywa każdy wierzchołek.

Linie 8 i 12 Podobieństwo operacji z linii 8 i 12 nie jest przypadkowe. Można na to patrzeć w następujący sposób: Δ jest obliczana w przypadku, kiedy nie została znaleziona ścieżka powiększająca o początku w u . Następstwem obliczenia tej wartości jest poszerzenie „zasięgu” wierzchołka u . Zbiór Z natomiast określa dotychczasowy jego „zasięg”. Widzimy więc, że zbiór Z możemy budować na bieżąco, *równocześnie* z poszukiwaniami ścieżki *path*. Nie ma potrzeby duplikowania tej operacji. W takiej wersji nie będziemy korzystać z algorytmu BFS, ale inkrementacyjnie budować graf DAG (jako podgraf grafu \vec{G}_p) o początku w u . Budujemy ten podgraf tak długo, aż znajdziemy ścieżkę powiększającą. Jeżeli nie jesteśmy w stanie dalej rozbudowywać tego grafu, a ścieżki wciąż nie znaleźliśmy – dopiero wtedy obliczamy Δ i powiększamy funkcję potencjału.

5.5.3 Pseudokod algorytmu $O(n^3)$

Na podstawie przedstawionych uwag, możemy stworzyć zarys działania szybszej wersji algorytmu węgierskiego. Do tego skorzystamy z pseudokodu 6. Nie jest to całość algorytmu, a jedynie jego kluczowa część, która powiększa licznosc skojarzenia o 1. Algorytm **Hungarian2** wykona się więc $O(n)$ razy i pokażemy, że jego złożoność wynosi $O(n^2)$.

Uwaga! Zbiory S i T są tutaj odpowiednikiem zbioru Z , jednak rozdzielonego ze względu na przynależność do klas dwudzielności.

Dla ułatwienia, w konkretnej implementacji będzie tworzona struktura danych (na przykład tablica), która dla każdego wierzchołka $y \notin T$ przechowuje wartość $\min\{w(x, y) - p(x) - p(y) : x \in S\}$. Korzystając z tablicy minimów, wartość Δ może zostać wyliczona w czasie $O(n)$. Wówczas aktualizacja tej samej tablicy może zostać wykonana również w czasie $O(n)$ przez ustawienie: $\forall_{y \in T} \text{tab}[y] = \text{tab}[y] + \Delta$. Złożoność linii **6-10** jest w takim razie liniowa.

Dodatkowym krokiem, niewyróżnionym w algorytmie, jest aktualizacja tablicy po linii **16**. To również zajmuje czas liniowy. Pozostałe kroki zajmują czas liniowy w sposób oczywisty.

Podsumowując, przedstawiliśmy, że faza 6 ulepszanego algorytmu węgierskiego wykonuje się w czasie $O(n^2)$, a sam algorytm węgierski składa się z co najwyżej $O(n)$ takich faz. W związku z czym sumaryczna jego złożoność czasowa wynosi $O(n^3)$.

Algorithm 6 Hungarian2

```

1:  $u \leftarrow$  dowolny wierzchołek z  $R_H$ 
2:  $S \leftarrow \{u\}$ 
3:  $T \leftarrow \emptyset$ 
4: while  $\text{path}$  is NULL do
5:   if  $N_p(S) = T$  then                                 $\triangleright$  Aktualizacja potencjału, wymuszenie  $N_p(S) \neq T$ 
6:      $\Delta \leftarrow \min\{w(s, y) - p(s) - p(y) : s \in S, y \notin T\}$ 
7:     for all  $v$  in  $S$  do  $p(v) \leftarrow p(v) + \Delta$ 
8:   end for
9:   for all  $v$  in  $T$  do  $p(v) \leftarrow p(v) - \Delta$ 
10:  end for
11: end if
12:  $y \leftarrow$  dowolny wierzchołek z  $N_p(S) - T$ 
13: if  $y$  - niepokryty przez  $M$  then
14:    $\text{path} \leftarrow$  ścieżka od  $u$  do  $y$                                  $\triangleright$  Przerwanie pętli while
15: else  $yz \in M$                                              $\triangleright$  Dalsze budowanie podgrafu DAG
16:    $S \leftarrow S \cup \{z\}$ 
17:    $T \leftarrow T \cup \{y\}$ 
18: end if
19: end while

```

6 Opis wejścia, wyjścia

W tej podsekcji przybliżono format plików stanowiących wejście oraz wyjście programu.

6.1 Wejście

Wejście programu stanowi plik zawierający następujące informacje:

- liczność zbioru reprezentującego studnie (n)
- liczba domów, do których może być doprowadzona woda przy użyciu jednej studni (k)
- lista zawierająca lokalizacje wszystkich studni
- lista zawierająca lokalizacje wszystkich domów

Wywołanie programu odbywa się poprzez uruchomienie pliku wykonywalnego i wskazanie ścieżki zawierającej instancję problemu jaki należy rozwiązać.

Przykład poprawnego wywołania oraz odpowiednio sformatowanego pliku stanowiącego wejście programu został przedstawiony na rysunku 1. Dopiski zamieszczone po znakach `/**` stanowią wyłącznie komentarze mające na celu objaśnienie formatu pliku wejściowego. Przy czytaniu pliku z właściwą instancją problemu ich umieszczenie spowoduje błąd programu wynikający z niewłaściwego formatu wejściowego.

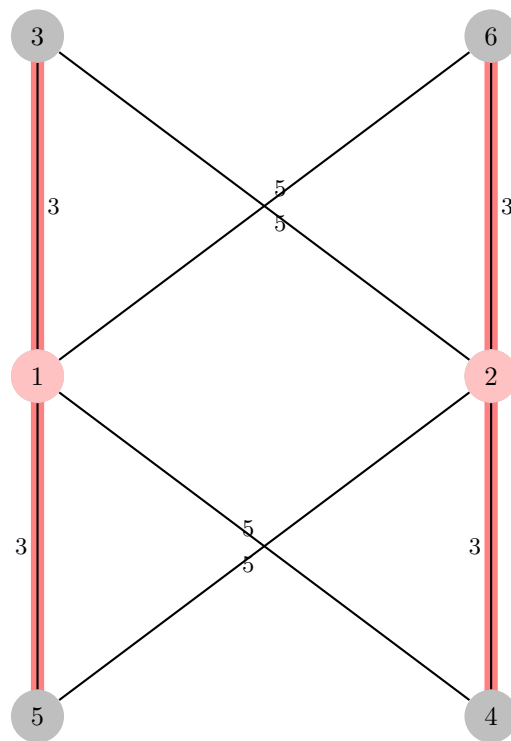
```

studnie_input - Notepad
File Edit Format View Help
2 // n - liczba studni
2 // k - liczba domów, które mogą zostać zaopatrzone w wodę przez pojedynczą studnię
1 2 0 // lokalizacje studni: [indeks wierzchołka] [współrzędna x] [współrzędna y]
2 6 0
1 2 3 // lokalizacje domów: [indeks wierzchołka] [współrzędna x] [współrzędna y]
2 6 -3
3 2 -3
4 6 3

```

Rysunek 1: Przykład pliku wejściowego

Poniższy rysunek przedstawia graf reprezentujący instancję problemu wczytaną w ramach odczytu pliku przedstawionego na grafice 1. Wierzchołki reprezentujące studnie zostały wyróżnione kolorem czerwonym. Na krawędziach naniesiono koszt dostarczenia wody do połączonego domu przez połączoną studnię. Ponadto kolorem czerwonym wyróżniono krawędzie odpowiadające optymalnemu przypisaniu domów do studni. W celu uzyskania jednoznaczności w interpretacji wierzchołków, numery indeksujące wierzchołki należące do zbioru domów zostały zwiększone o licznosc zbioru reprezentującego studnie (w wyniku tej operacji pierwszy dom uzyskał indeks o wartości trzy itd.).

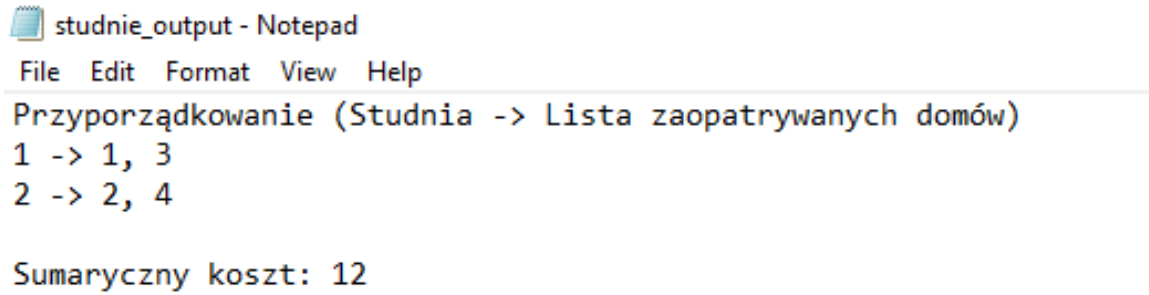


6.2 Wyjście

Wyjście programu stanowi plik zawierający następujące informacje:

- lista zawierająca kolejne studnie oraz zaopatrywane przez nią domy
- minimalny sumaryczny koszt dostarczenia wody do każdego domu (rozumiany jako suma odległości euklidesowych wszystkich połączeń)

Plik stanowiący rozwiązanie rozpatrywanej instancji problemu jest zapisywany pod taką samą nazwą jak plik wejściowy z sufiksem `_output`. Ponadto rozwiązanie problemu jest również wypisywane w konsoli. Przykład pliku stanowiącego wyjście problemu został przedstawiony na rysunku 2.



```
studnie_output - Notepad
File Edit Format View Help
Przyporządkowanie (Studnia -> Lista zaopatrywanych domów)
1 -> 1, 3
2 -> 2, 4

Sumaryczny koszt: 12
```

Rysunek 2: Przykład pliku wyjściowego

Literatura

- [1] S. Giancola. *Bipartite Matching : Framework and Solutions*. 2022.
- [2] M. Golin. *Bipartite Matching the Hungarian Method*. 2006.
- [3] G. Righin. *Minimum cost bipartite matching*.
- [4] Tim Roughgarden. *CS261: A Second Course in Algorithms, Lecture 5: Minimum-Cost Bipartite Matching*. 2016.