Sprawozdanie Lista 4

Tymoteusz Roźmiarek

12 stycznia 2025

Maksymalny przepływ

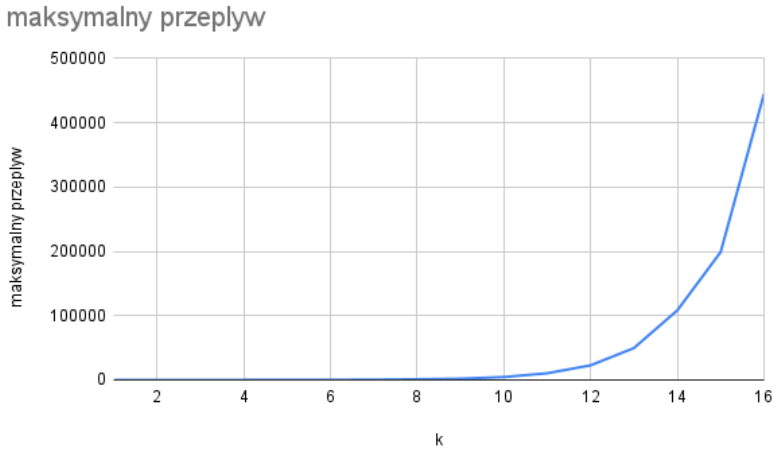
Sieć rezydualna to graf reprezentujący, ile dodatkowego przepływu można przesłać przez sieć przepływową, biorąc pod uwagę istniejący przepływ. Dla krawędzi z przepływem, pojemność rezydualna jest różnicą między oryginalną pojemnością a aktualnym przepływem.
Ścieżka powiększająca to ścieżka w sieci rezydualnej z węzła źródłowego do węzła ujścia, wzdłuż której można zwiększyć przepływ. Zwiększenie przepływu wzdłuż ścieżki powiększającej oznacza dodanie pewnej wartości do przepływu na każdej krawędzi tej ścieżki.

Algorytm Edmondsa-Karpa jest odmianą algorytmu Forda-Fulkersona do znajdowania maksymalnego przepływu w sieci przepływowej. Działa on poprzez wielokrotne znajdowanie ścieżki powiększającej w sieci rezydualnej i zwiększanie przepływu wzdłuż tej ścieżki. W przeciwieństwie do ogólnego algorytmu Forda-Fulkersona, algorytm Edmondsa-Karpa zawsze wybiera najkrótszą ścieżkę powiększającą, gdzie długość jest mierzona liczbą krawędzi. To ograniczenie gwarantuje, że algorytm zakończy się w czasie wielomianowym. Algorytm można podsumować następująco:

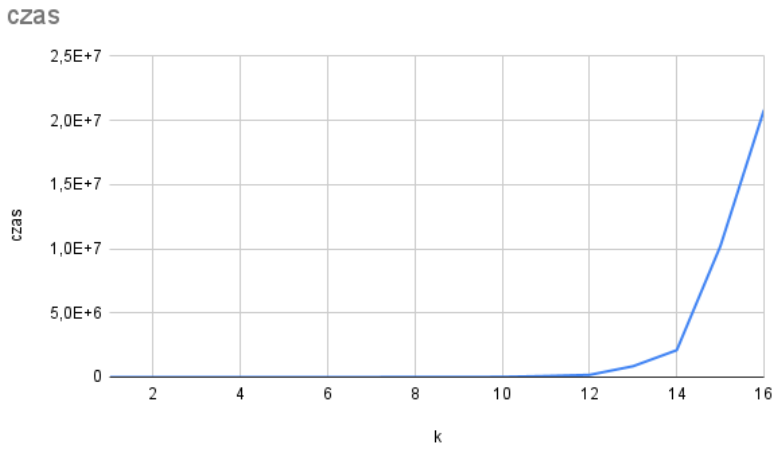
1. Inicjalizacja: Ustaw przepływ na każdej krawędzi na 0.
2. Pętla: Dopóki istnieje ścieżka powiększająca w sieci rezydualnej:
    - Znajdź najkrótszą ścieżkę powiększającą za pomocą przeszukiwania wszerz (BFS).
    - Zwiększ przepływ wzdłuż tej ścieżki o maksymalną możliwą wartość.
    - Zaktualizuj sieć rezydualną.
3. Zwróć znaleziony przepływ.

Algorytm Edmondsa-Karpa ma złożoność czasową $O(nm^2)$, gdzie n to liczba węzłów, a m to liczba krawędzi w sieci.

Wyniki



Rysunek 1: Maksymalny przepływ w zależności od k



Rysunek 2: Czas działania programu w zależności od k (w mikrosekundach)

Rysunek 3: Liczba ścieżek powiększających w zależności od k

Skojarzenie o największym rozmiarze

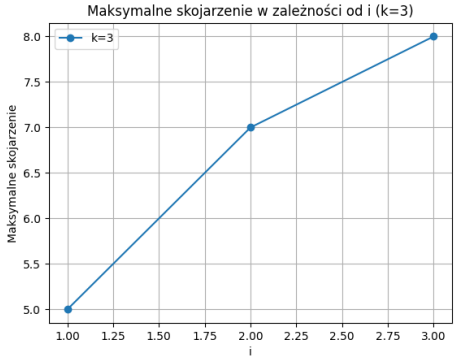
Do rozwiązania wykorzystałem **algorytm Hopcroft-Karp** . Jest to ulepszenie w stosunku do algorytmu Edmondsa-Karpa. W przypadku grafów dwudzielnych, problem maksymalnego dopasowania można przekształcić w problem maksymalnego przepływu, ale algorytm Hopcroft-Karp działa szybciej. Algorytm Hopcroft-Karp opiera się na koncepcji sieci rezydualnej i ścieżek powiększających, podobnie jak algorytm Edmondsa-Karpa. W sieci rezydualnej dla grafu dwudzielnego, każda ścieżka powiększająca ma długość co najmniej 2 (ponieważ musi przechodzić przez co najmniej jeden węzeł z każdego "zbioru" w grafie dwudzielnym). Algorytm Hopcroft-Karp wykorzystuje ten fakt, aby w każdej fazie znajdować maksymalny zbiór ścieżek powiększających o minimalnej długości, które nie mają ze sobą wspólnych węzłów. Zwiększa on przepływ wzdłuż wszystkich ścieżek w tym zbiorze jednocześnie, co przyspiesza proces znajdowania maksymalnego dopasowania.
Algorytm można podsumować następująco:

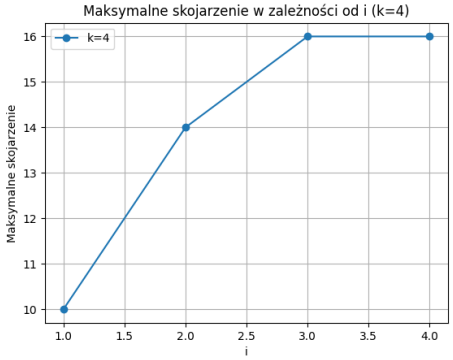
1. Inicjalizacja: Ustaw dopasowanie na puste.
2. Pętla: Dopóki istnieje ścieżka powiększająca w sieci rezydualnej:
   - Znajdź maksymalny zbiór rozłącznych ścieżek powiększających o minimalnej długości.
   - Zwiększ dopasowanie o wszystkie krawędzie w znalezionych ścieżkach.
   - Zaktualizuj sieć rezydualną.
3. Zwróć maksymalne skojarzenie.

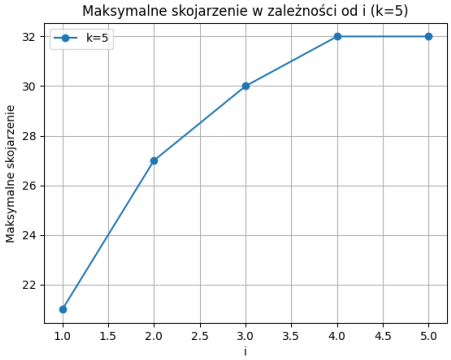
Algorytm Hopcroft-Karp ma złożoność czasową $O(\sqrt{n}m)$, gdzie n to liczba węzłów, a m to liczba krawędzi w grafie.

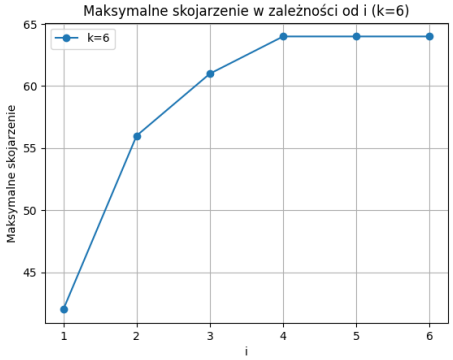
Wyniki

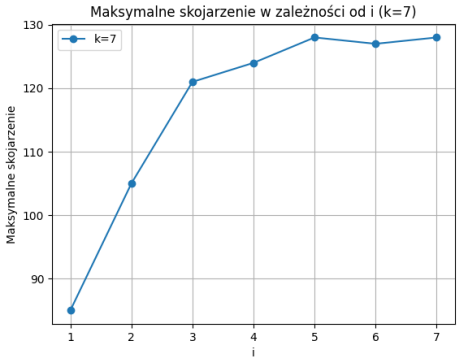
Wykresy wielkości maksymalnego skojarzenia w zaleznosci od i, dla kazdego k

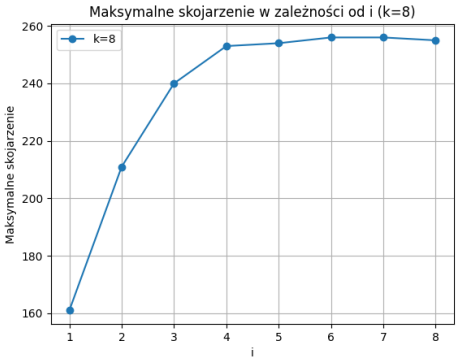


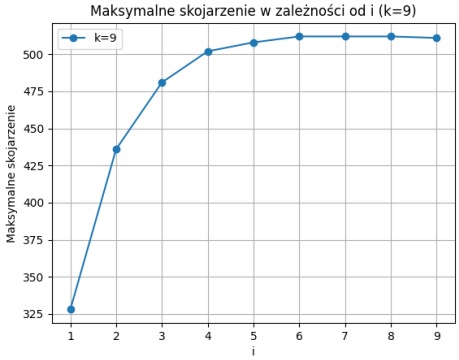


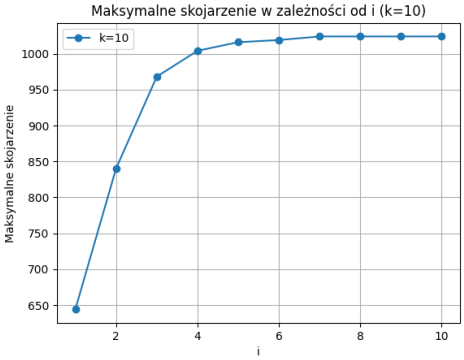




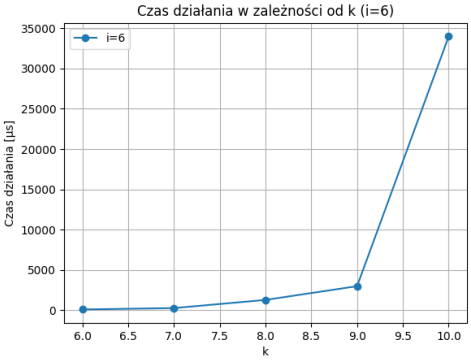


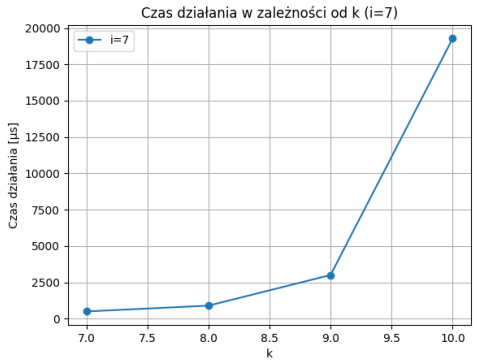


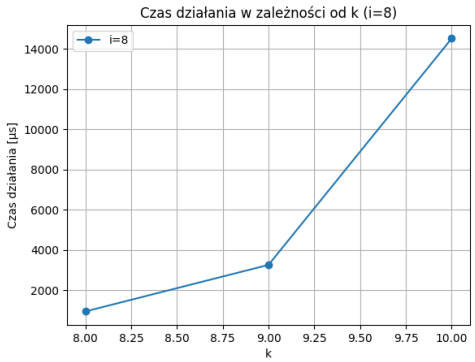




Wykresy wykresy czasu działania programu w zależności od k, dla każdego i







Modele programowania liniowego

Model dla zadania znajdowania maksymalnego przepływu:

Zbiór wierzchołków:	V
Zbiór krawędzi:	$E \subseteq V \times V$
Pojemność krawędzi:	$c_{ij} \geqslant 0, \quad \forall (i,j) \in E$
Wierzchołek źródłowy:	$s \geqslant 0$
Wierzchołek ujścia:	$t > 0$
Przepływ po krawędzi:	$f_{ij} \geqslant 0, \quad \forall (i,j) \in E$

Problem maksymalnego przepływu:

$$\begin{aligned} &\text{maksymalizuj} && \sum_{(s,j)\in E} f_{sj} \\ &\text{pod warunkiem że} && f_{uv} \leqslant c_{uv}, \quad \forall (u,v) \in E \\ & && \sum_{u\in V:(u,v)\in E} f_{uv} = \sum_{w\in V:(v,w)\in E} f_{vw}, \quad \forall v \in V \setminus \{s,t\} \end{aligned}$$

Wyniki

Tabela 1: Porównanie wyników algorytmu C++ i solvera dla $k = 2$

Parametr	C++ (s)	Solver (s)
Maksymalny przepływ	5	5
Czas (s)	0.0	0.0

Tabela 2: Porównanie wyników algorytmu C++ i solvera dla $k = 7$

Parametr	C++ (s)	Solver (s)
Maksymalny przepływ	339	339
Czas (s)	0.0	0.0

Tabela 3: Porównanie wyników algorytmu C++ i solvera dla $k = 13$

Parametr	C++ (s)	Solver (s)
Maksymalny przepływ	44555	44555
Czas (s)	0.809	7.8

Tabela 4: Przepływy po krawędziach C++ i Solver dla $k = 2$

Krawędź	Przepływ (C++)
$0 \to 1$	1
$0 \to 2$	2
$1 \to 3$	1
$2 \to 3$	2

Krawędź	Przepływ (Solver)
$0 \to 1$	1
$0 \to 2$	2
$1 \to 3$	1
$2 \to 3$	2

Tabela 5: Przepływy po krawędziach C++ i Solver dla $k = 3$

Krawędź	Przepływ (C++)
$0 \to 1$	3
$0 \to 2$	2
$0 \to 4$	3
$1 \to 3$	3
$1 \to 5$	0
$2 \to 3$	0
$2 \to 6$	2
$3 \to 7$	3
$4 \to 5$	2
$4 \to 6$	1
$5 \to 7$	2
$6 \to 7$	3

Krawędź	Przepływ (Solver)
$0 \to 1$	3
$0 \to 2$	2
$0 \to 4$	3
$1 \to 3$	3
$1 \to 5$	0
$2 \to 3$	0
$2 \to 6$	2
$3 \to 7$	3
$4 \to 5$	2
$4 \to 6$	1
$5 \to 7$	2
$6 \to 7$	3

Model dla zadania znajdowania największego skojarzenia:

Zbiory wierzchołków:	U, V
Macierz sąsiedztwa:	$e_{uv} \in \{0, 1\}, \quad \forall u \in U, v \in V$
Zmienna decyzyjna:	$x_{uv} \in \{0, 1\}, \quad \forall u \in U, v \in V$

Problem maksymalnego skojarzenia:

$$\begin{aligned} \text{maksymalizuj} \quad & \sum_{u \in U} \sum_{v \in V} x_{uv} \\ \text{pod warunkiem że} \quad & \sum_{v \in V} x_{uv} \leqslant 1, \quad \forall u \in U \\ & \sum_{u \in U} x_{uv} \leqslant 1, \quad \forall v \in V \\ & x_{uv} \leqslant e_{uv}, \quad \forall u \in U, v \in V \end{aligned}$$

Wyniki

Tabela 6: Porównanie wyników algorytmu C++ i solvera dla $k = 4,, i = 2$

Parametr	C++ (s)	Solver (s)
Maksymalne skojarzenie	13	13
Czas (s)	0.0	0.0

Tabela 7: Porównanie wyników algorytmu C++ i solvera dla $k = 8, i = 4$

Parametr	C++ (s)	Solver (s)
Maksymalny przepływ	248	248
Czas (s)	0.003	0.7

Tabela 8: Porównanie wyników algorytmu C++ i solvera dla $k = 10, i = 5$

Parametr	C++ (s)	Solver (s)
Maksymalny przepływ	1016	1016
Czas (s)	0.025	82.8

C++	Solver
(0, 1)	(0, 1)
(1, 0)	(1, 0)
(3, 2)	(3, 2)

Tabela 9: $k = 2, i = 1$

C++	Solver
(0, 1)	(0, 1)
(1, 6)	(1, 6)
(2, 3)	(2, 3)
(3, 0)	(3, 0)
(4, 4)	(4, 4)
(6, 7)	(6, 7)
(7, 5)	(7, 5)

Tabela 10: $k = 3, i = 2$

C++	Solver
(0, 4)	(0, 4)
(1, 2)	(1, 2)
(2, 3)	(2, 3)
(3, 13)	(3, 13)
(4, 5)	(4, 5)
(6, 11)	(6, 11)
(7, 0)	(7, 0)
(8, 8)	(8, 8)
(9, 10)	(9, 10)
(10, 6)	(10, 6)
(11, 9)	(11, 9)
(12, 7)	(12, 7)
(13, 12)	(13, 12)
(15, 1)	(15, 1)

Tabela 11: $k = 4, i = 2$

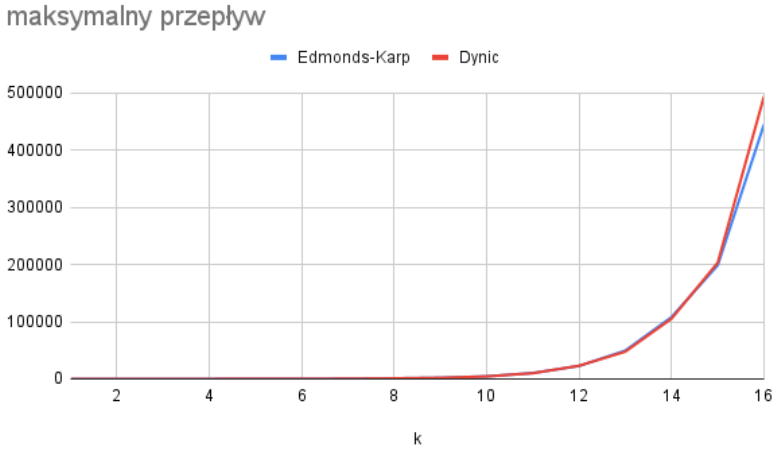
Algorytm Dynica

Sieć warstwowa składa się z warstw węzłów, gdzie warstwa 0 to węzeł źródłowy, a warstwa k to zbiór węzłów, do których można dotrzeć z węzła źródłowego w dokładnie k krokach w sieci rezydualnej.
Algorytm Dynica jest algorytmem służącym do znajdowania maksymalnego przepływu w sieci przepływowej. Działa on poprzez budowanie sieci warstwowej, która jest podgrafem sieci rezydualnej, i znajdowanie w niej ścieżek powiększających. Algorytm Dynica wyszukuje ścieżki powiększające w sieci warstwowej, dopóki nie będzie już możliwe dotarcie do węzła ujścia. Wtedy buduje nową sieć warstwową i powtarza proces.
Algorytm można podsumować następująco:

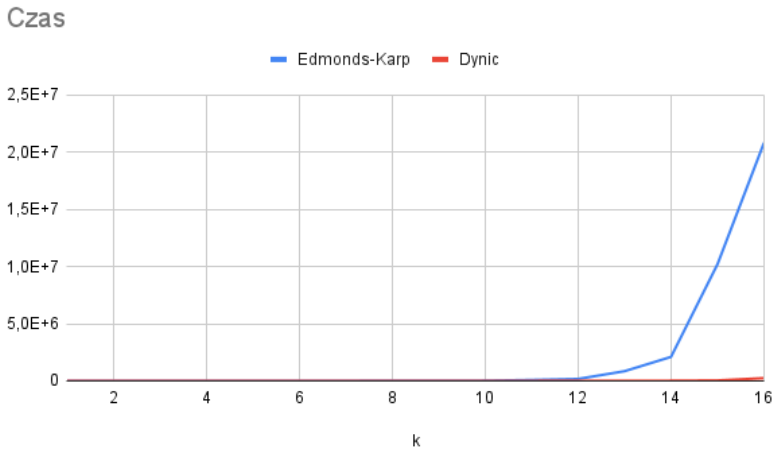
1. Inicjalizacja: Ustaw przepływ na każdej krawędzi na 0.
2. Pętla: Dopóki istnieje ścieżka z węzła źródłowego do węzła ujścia w sieci rezydualnej:
   - Zbuduj sieć warstwową.
   - Znajdź ścieżki powiększające w sieci warstwowej i zwiększ przepływ wzdłuż tych ścieżek.
3. Zwróć znaleziony przepływ.

Algorytm Dynica ma złożoność czasową $O(n^2m)$, gdzie n to liczba węzłów, a m to liczba krawędzi w sieci. W praktyce algorytm Dynica często działa szybciej, szczególnie w przypadku gęstych sieci.

Wyniki



Rysunek 4: Maksymalny przepływ w zależności od k



Rysunek 5: Czas działania programu w zależności od k (w mikrosekundach)

Rysunek 6: Liczba ścieżek powiększających w zależności od k

Na powyższych wykresach widać, że algorytm Dynica działa o wiele szybciej niż algorytm Edmondsa-Karpa. Różnica robi się bardzo zauważalna dla $k \geqslant 12$. Ponadto algorytm Dynica wykorzystuje zdecydowanie więcej ścieżek powiększających.