Algorytmy optymalizacji dyskretnej, lista 4

Adrian Wilhelmi, 268479

6 stycznia 2025 r.

Zadanie 1

Strutura danych oraz algorytm

Celem zadania jest zaprojektowanie struktury danych reprezetnującej k-wymiarową hiperkostkę jako graf skierowany oraz zaimplementowanie algorytmu Edmondsa-Karpa w celu znalezienia maksymalnego przepływu w tym grafie. Poniżej zaprezentowano zaprojektowaną strukturę.

---

1 **Class *Graph*:**
2 `k` – wymiar hiperkostki;
3 `num_nodes` – liczba wierzchołków;
4 `num_edges` – liczba krawędzi;
5 `adj_list` – lista sąsiedztwa;

Konstruktor klasy *Graph* generuje hiperkostkę o podanym wymiarze k zgodnie z treścią zadania oraz w odpowiedni sposób losowo inicializuje wartości maksymalnego przepływu dla każdej krawędzi.

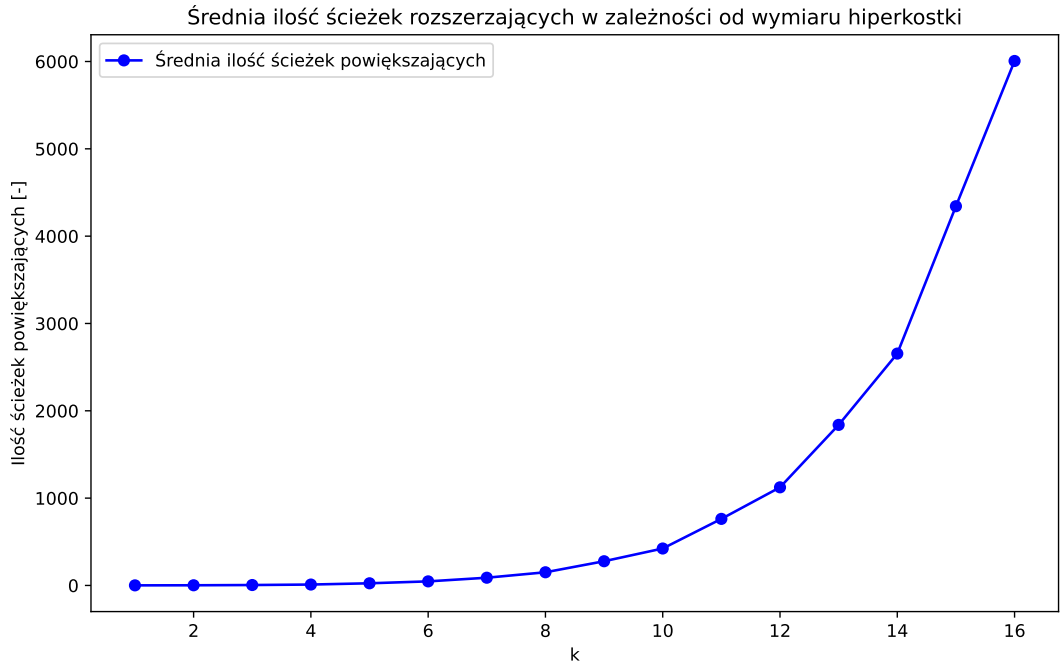
Złożoność czasowa

Zaimplementowano algorytm Edmondsa-Karpa bazujący na metodzie Forda-Fulkersona używając BFS do znajdowania najkrótszej ścieżki powiększającej w grafie rezydualnym. Gwarantuje to złożoność $O(|V||E|^2)$. Z racji, że $|V| = 2^k$ oraz $|E| = k2^{k-1}$ otrzymujemy złożoność w zależności od wymiaru kostki k:

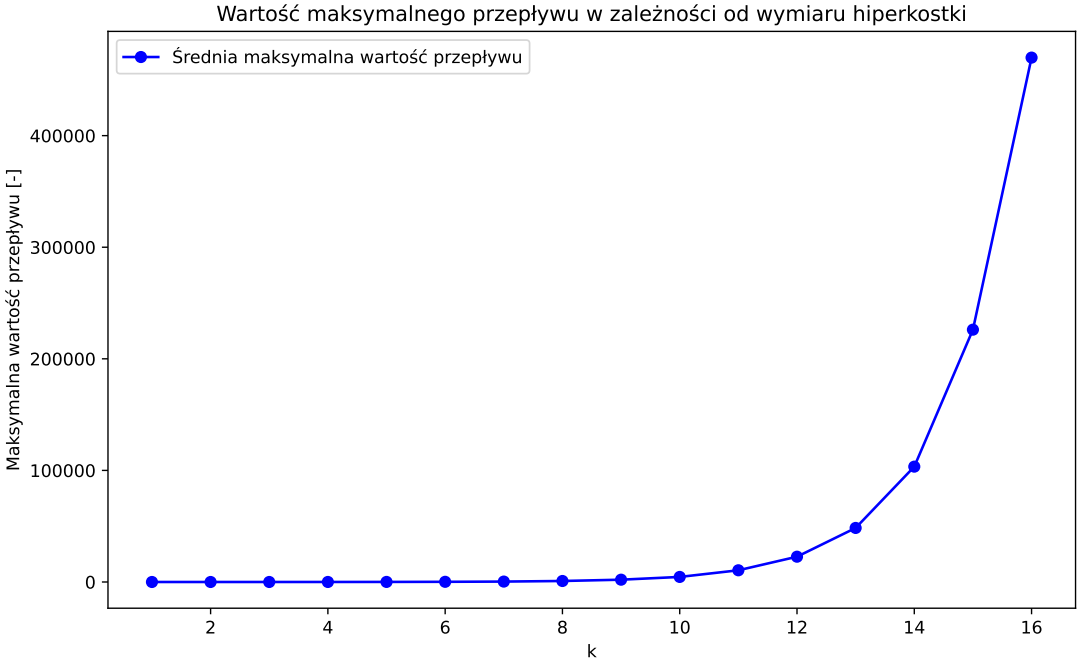
$$O(|V||E|^2) = O(2^k \cdot (k2^{k-1})^2) = O(2^k \cdot k^2 \cdot 2^{2k-2}) = O(k^2 \cdot 2^{3k2}) = O(k^2 2^k)$$

Wyniki oraz wnioski

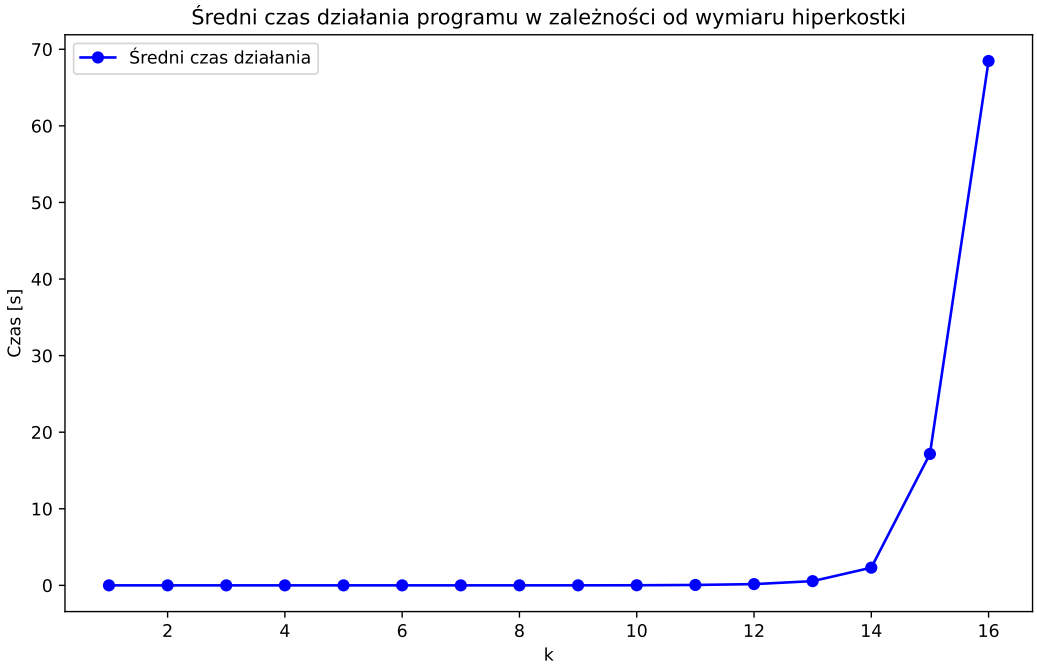
Wyniki otrzymane empirycznie przedstawiono na Rysunkach 1, 2 oraz 3. Dla każdego k wykoanno 50 niezależnych powtórzeń eksperymentów i na wykresach zapisano uśrednioną wartość. Wyniki z wykresów 1 oraz 3 potwierdzają eksponencjalny wzrost czasu wykonania algorytmu w zalezności od wymiaru hiperkostki k. Wartość maksymalnego przepływu również wzrasta eksponencjalnie jak widać na Rysunku 2, co ma uzasadnienie biorąc pod uwagę sposób losowania pojemności krawędzi – pojemności są losowane z wykładniczo rosnącego zakresu (2^l) i ponieważ liczba możliwych ścieżek między źródłem a ujściem rośnie również wykładniczo, średnia maksymalna wartość przepływu staje się proporcjonalna do 2^k.



Rysunek 1: Średnia liczba ścieżek powiększających w zależności od wymiaru hiperkostki



Rysunek 2: Średnia wartość maksymalnego przepływu w zależności od wymiaru hiperkostki



Rysunek 3: Średnia liczba ścieżek powiększających w zależności od wymiaru hiperkostki

Zadanie 2

Algorytm Kuhna

Celem zadania jest zbadanie eksperymentalnie wielkości maksymalnego skojarzenia w losowym grafie dwudzielnym $G = (V_1 \cup V_2, E)$, o zadanym $|V_1| = |V_2| = 2^k$ oraz stopniu wierzchołków i ze zbioru V_1. Zaimplementowano w tym celu algorytm Kuhna, który używając DFS dla każdego wierzchołka z V_1 szuka połączenia z wierzchołkiem V_2 przepinając ewentualnie wcześniej przypisane połączenia w przypadku kolizji. Z każdym przypisaniem zwiększany jest licznik skojarzeń.

Złożoność czasowa

DFS wywoływany jest na każdym wierzchołku z V_1. W najgorszym wypadku dla każdego $v \in V_1$ odwiedzony zostanie każdy z jego sąsiadów, co wynosi i. $|V_1| = 2^k$, co daje łącznie złożoność

$$O(i2^k).$$

Wyniki oraz wnioski

Na Rysunkach od 4 do 11 obserwowalny jest logarytmiczny wzrost wartości maksymalnego skojarzenia w zależności od stopnia wierzchołków ze zbioru V_1 – intuicyjnie wzrost stopnia ogranicza dostępność wierzchołków z V_2 co skutkuje spowolnionym tempem wzrostu wartości maksymalnego skojarzenia. Na Rysunkach od 12 do 21 zauważyć można eksponencjalny wzrost czasu wykonania algorytmu w zależności od parametru k, co jest zgodne z teoretycznymi obliczeniami.

Zadanie 3

Dla problemów zdefiniowanych w Zadaniach 1 oraz 2 napisano równoważne modele programowania liniowego. Model w pakiecie $JuMP$ z języka $Julia$ zostanie wygenerowany dla programu po włączeniu go z odpowiednią komendą.

Model dla zadania 1

$$\begin{aligned} G = (V, E) &- \text{wejściowy graf} \\ s \in V &- \text{źródło} \\ t \in V &- \text{ujście} \\ x_{i,j} &- \text{przepływ dla krawędzi (i, j)} \\ c_{i,j} &- \text{pojemność krawędzi (i, j)} \end{aligned}$$

$x_{i,j}$ jest zmienną decyzyjną. Ograniczenia:

$$(\forall i, j)(c_{i,j} \geqslant x_{i,j} \geqslant 0)$$

$$(\forall i \in V \setminus (\{s, t\}))(\sum_{(i,j)\in E} x_{i,j} = \sum_{(j,i)\in E} x_{j,i})$$

Funkcja celu:

$$\max \sum_{(s,i)\in E} x_{s,i}$$

Model dla zadanie 2

$$\begin{aligned} G = (V_1 \cup V_2, E) &- \text{wejściowy graf} \\ x_{i,j} &- \text{Czy krawędź (i, j)} \in E \text{ należy do skojarzenia} \end{aligned}$$

$x_{i,j}$ jest binarną zmienną decyzyjną. Ograniczenia:

$$(\forall i \in V_1)(\sum_{(i,j)\in E} <= 1)$$

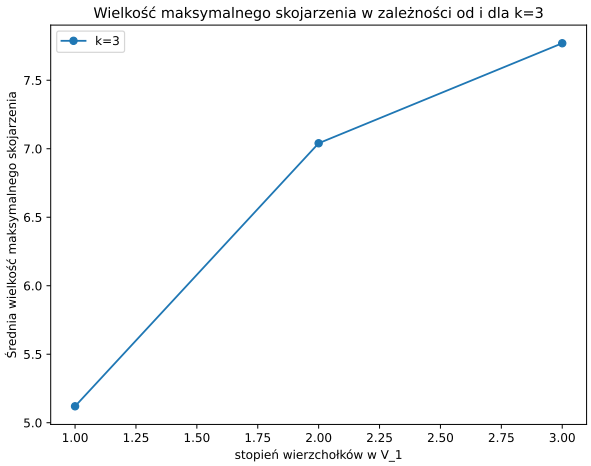
$$(\forall j \in V_2)(\sum_{(i,j)\in E} <= 1)$$

Funkcja celu:

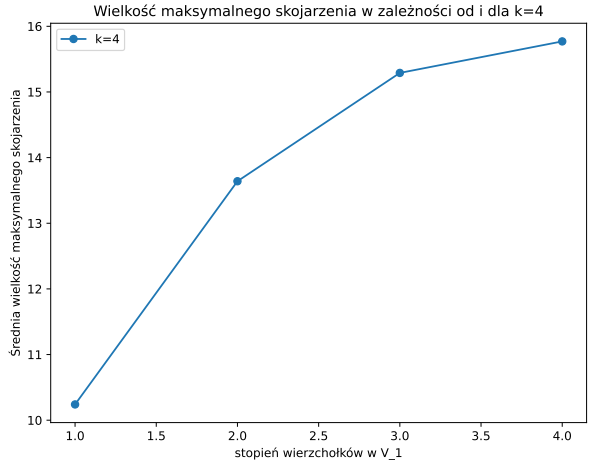
$$\max \sum_{(i,j)\in E} x_{i,j}$$

Obserwacje oraz wnioski

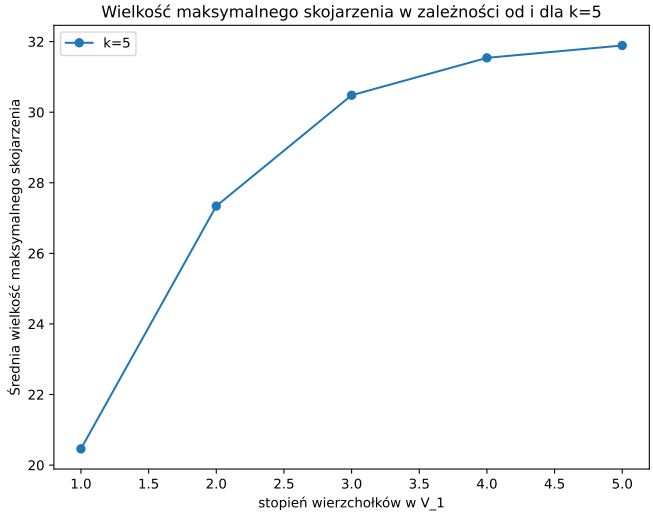
Modele LP zwracają te same wartości maksymalnego przepływu oraz wielkości maksymalnego skojarzenia — implikuje to poprawność zaimplementowanych algorytmów oraz modeli. Algorytmy zaimplementowane w C++ znajdują rozwiązania szybciej niż modele LP, co ma sens, ponieważ C++ pozwala na większą kontrole pamięci, a JuMP nie jest zoptymalizowany pod kątem szybkości obliczeń. Rozwiązania zwrócone przez algorytmy napisane w C++ różnią się minimalnie od tych zwracanych przez modele LP, co sugeruje różne podejście w obliczaniu przepływów oraz skojarzeń w grafie.



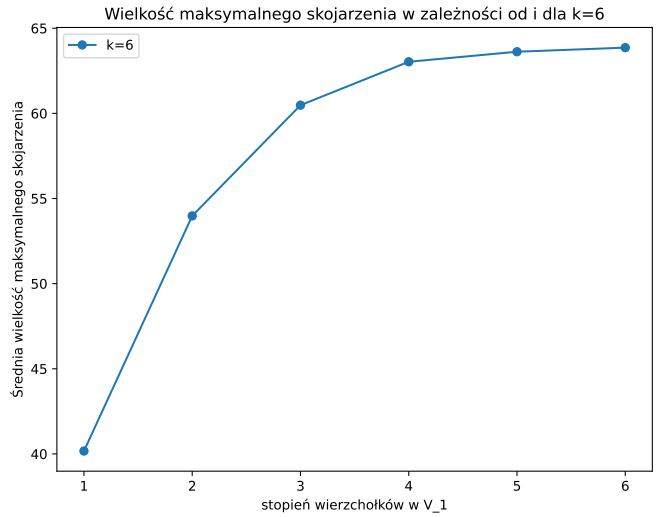
Rysunek 4: Wielkość maksymalnego skojarzenia w zalezności od i dla k=3



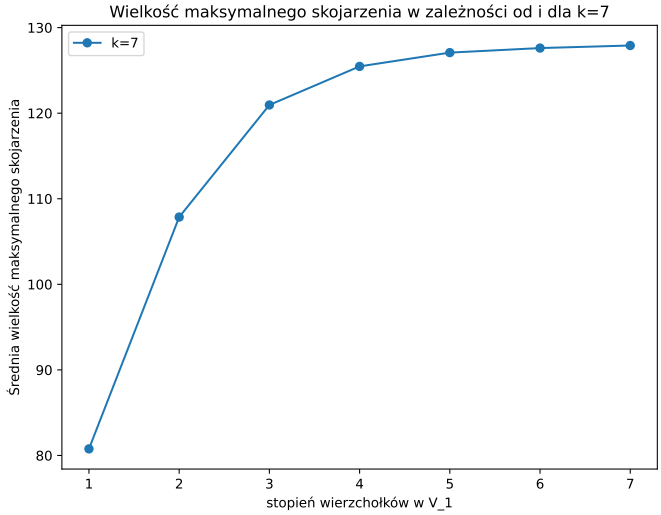
Rysunek 5: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 4$



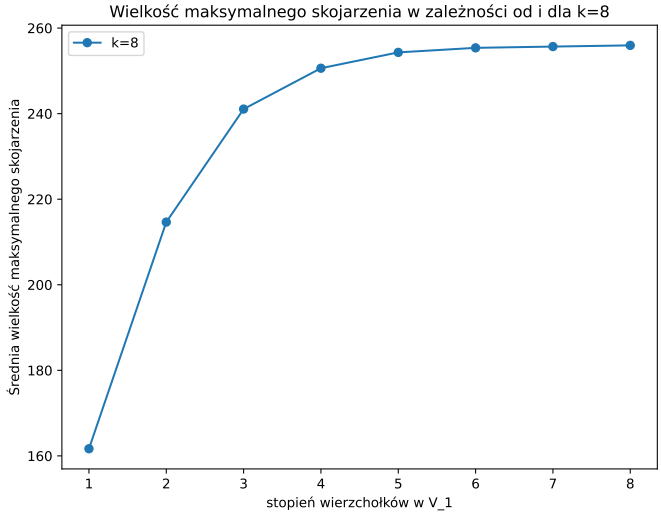
Rysunek 6: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 5$



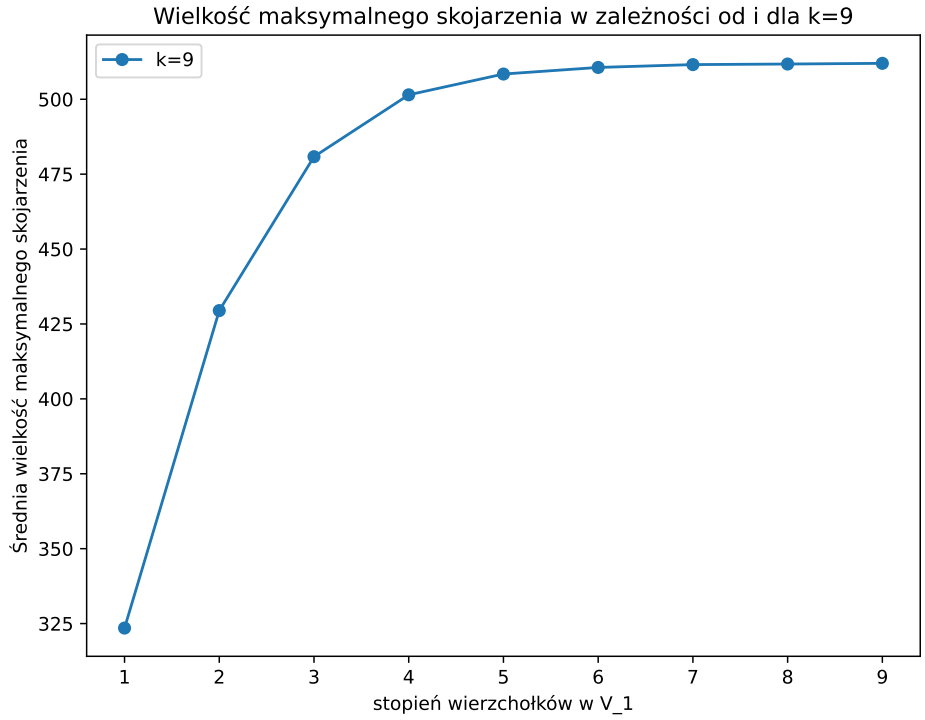
Rysunek 7: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 6$



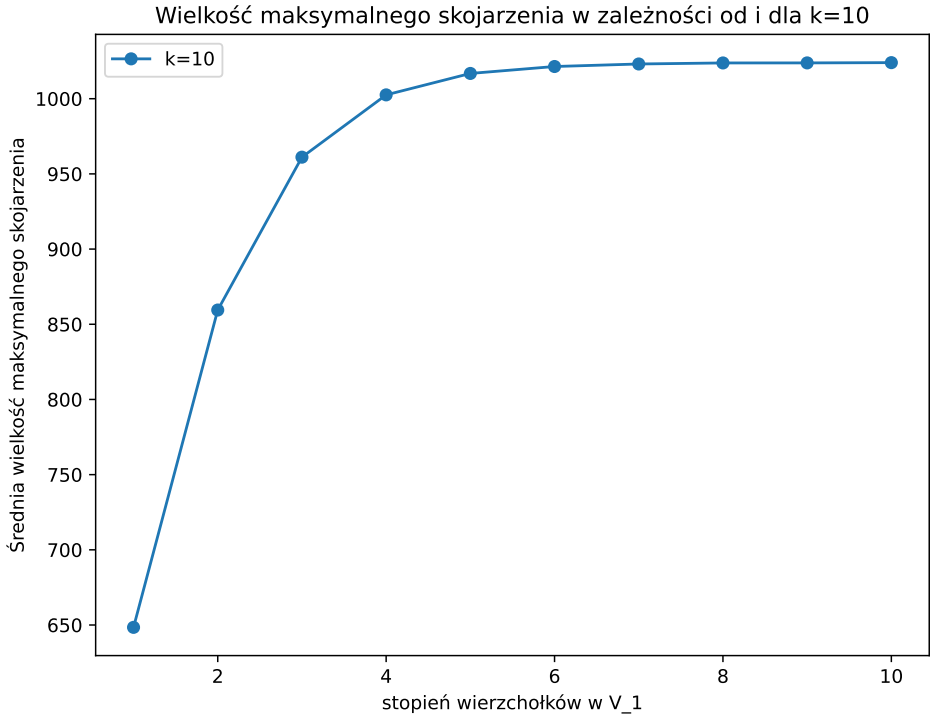
Rysunek 8: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 7$



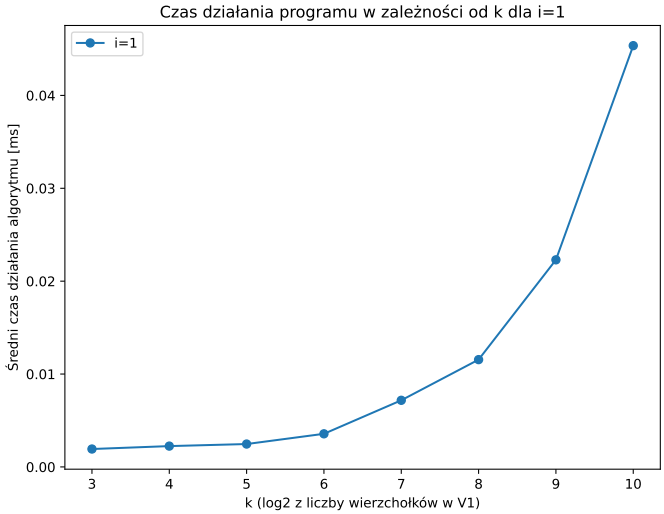
Rysunek 9: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 8$



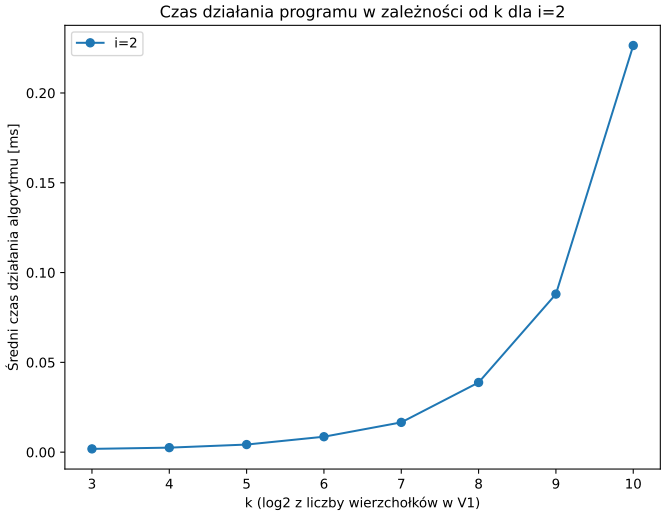
Rysunek 10: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 9$



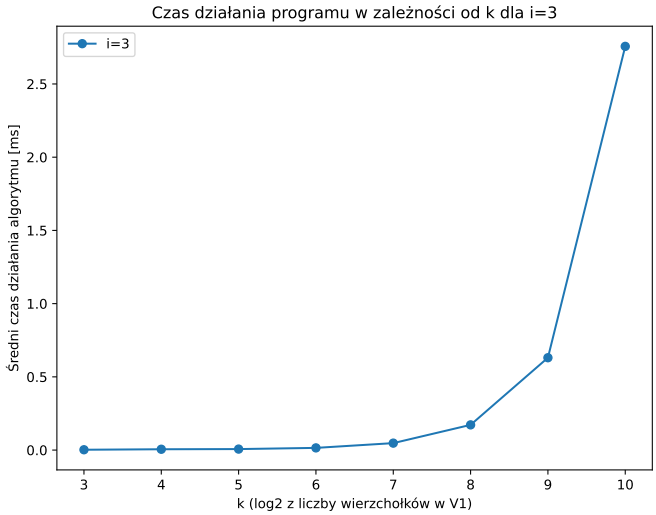
Rysunek 11: Wielkość maksymalnego skojarzenia w zależności od i dla $k = 10$



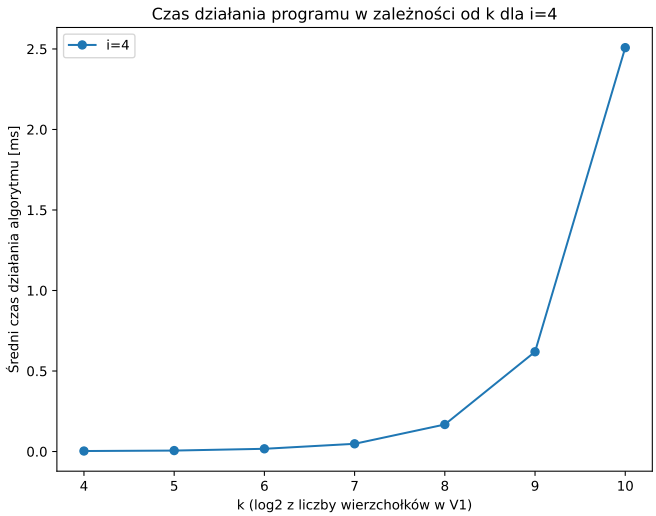
Rysunek 12: Czas wykonania algorytmu w zależności od i dla $k = 1$



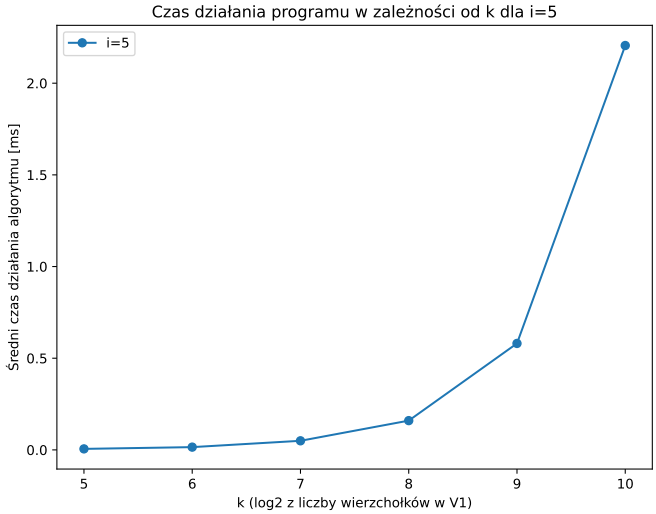
Rysunek 13: Czas wykonania algorytmu w zależności od i dla $k = 2$



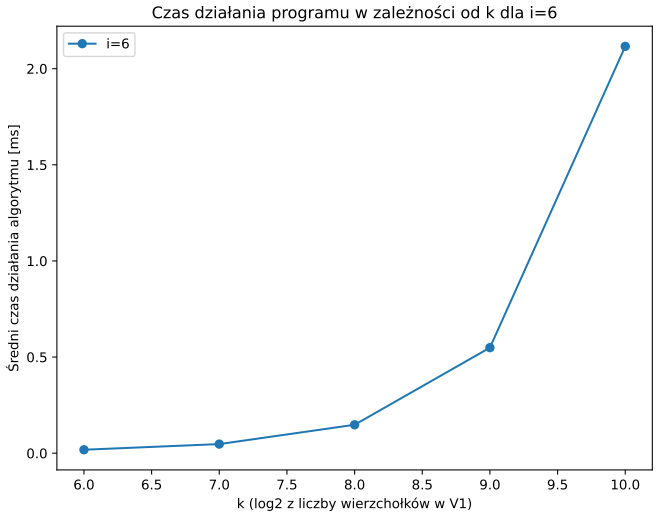
Rysunek 14: Czas wykonania algorytmu w zależności od i dla $k = 3$



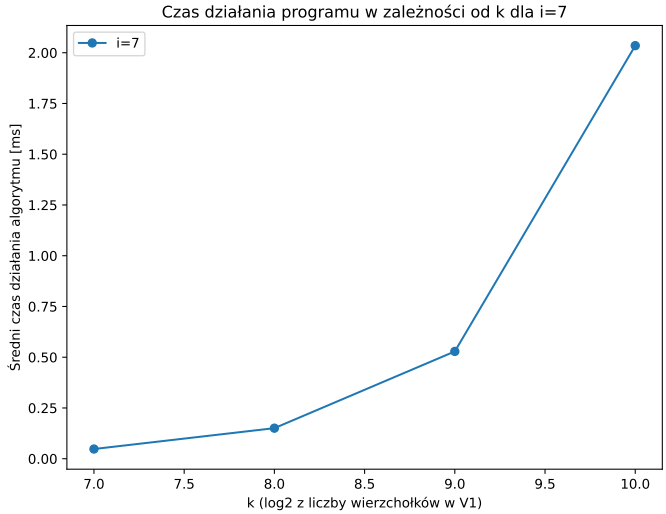
Rysunek 15: Czas wykonania algorytmu w zależności od i dla $k = 4$



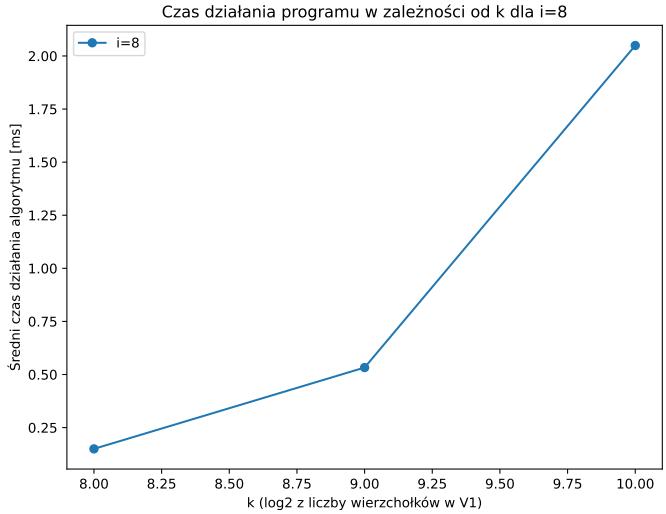
Rysunek 16: Czas wykonania algorytmu w zależności od i dla $k = 5$



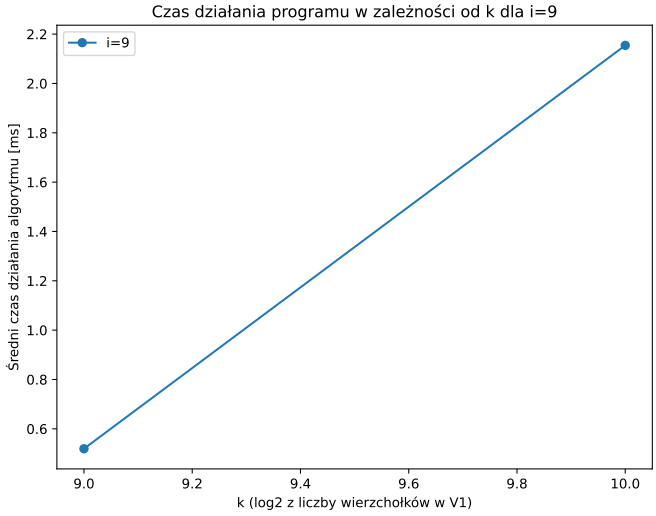
Rysunek 17: Czas wykonania algorytmu w zależności od i dla $k = 6$



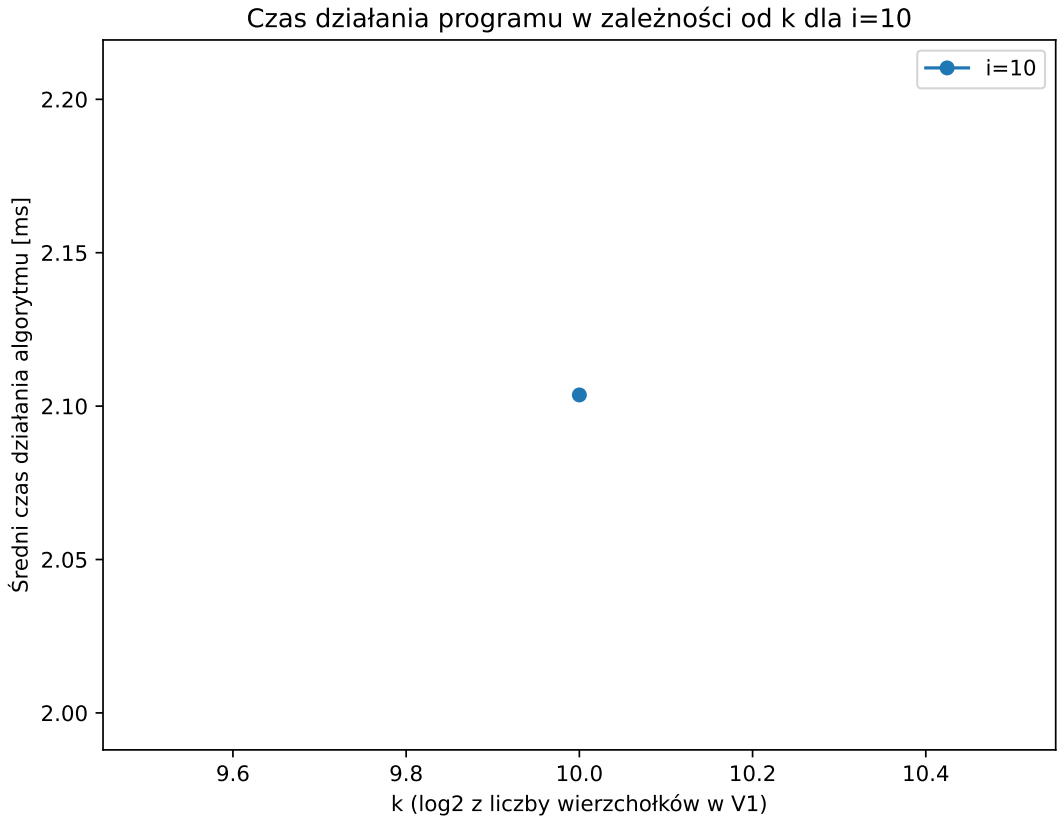
Rysunek 18: Czas wykonania algorytmu w zależności od i dla $k = 7$



Rysunek 19: Czas wykonania algorytmu w zależności od i dla $k = 8$



Rysunek 20: Czas wykonania algorytmu w zależności od i dla $k = 9$



Rysunek 21: Czas wykonania algorytmu w zależności od i dla $k = 10$