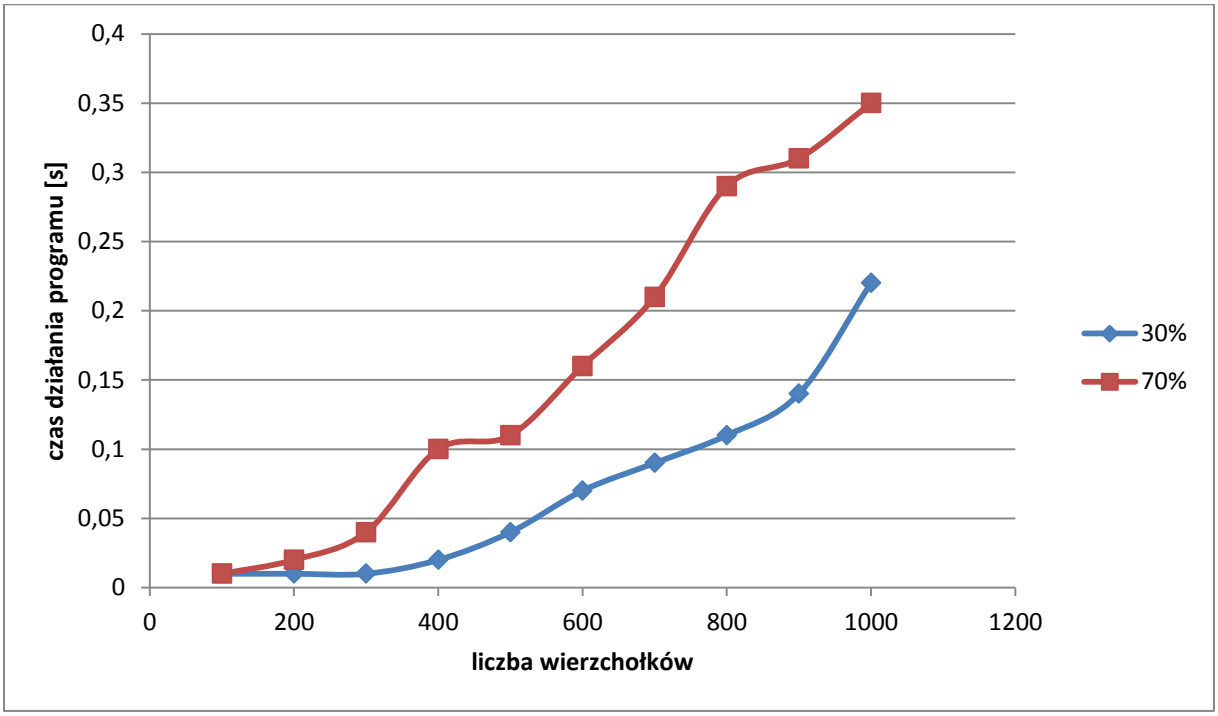
Algorytmy z powracaniem – sprawozdanie

Cykl Eulera



Złożoność:

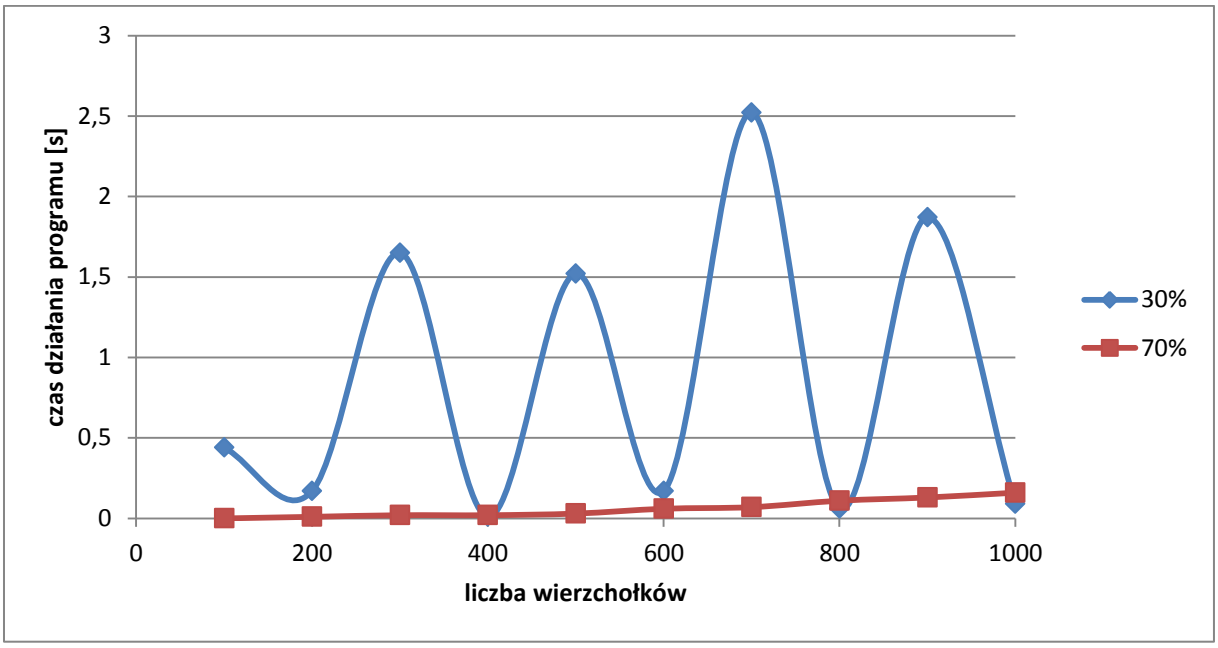
- **O(m) – dla listy następników**

Cykl Eulera to taki cykl w grafie, który przechodzi przez każdą jego krawędź dokładnie raz. Nie w każdym grafie można jednak go znaleźć. Aby istniała taka możliwość, graf musi być przede wszystkim spójny. Musi spełniać też jeszcze dodatkowy warunek zależny od tego, czy mamy określony kierunek przechodzenia przez jego krawędzie. Jeżeli graf jest nieskierowany jego wierzchołki muszą mieć stopień parzysty, a jeżeli jest skierowany stopień wejściowy wierzchołka musi być równy wyjściowemu.

Algorytm wyszukiwania cyklu Eulera opiera się na algorytmie przeszukiwania grafu w głąb. Graf przechodzimy metodą DFS, lecz zamiast sprawdzać, przed wywołaniem rekurencyjnej funkcji, czy wierzchołek był już odwiedzony usuwamy krawędź, którą przyszliśmy do niego. Po zakończeniu przetwarzania danego wierzchołka umieszczamy go na początku kolejki. Na końcu w kolejce powinno się znaleźć m+1 wierzchołków (gdzie m to liczba krawędzi), które odwiedzane po kolei spowodują przejście przez każdą krawędź grafu. Algorytm ten ma zatem złożoność O(m) pod warunkiem, że mamy natychmiastowy dostęp do następników danego wierzchołka i nie tracimy dodatkowego czasu na ich wyszukiwanie. Z tego powodu zaimplementowaliśmy ten algorytm, wykorzystując listę następników, które daje możliwość bezzwłocznego znalezienia następników.

Na wykresie widać, że cykl Eulera był znajdowany szybciej, gdy gęstość grafu była mniejsza. Jest to całkiem zrozumiałe, ponieważ złożoność algorytmu zależy bezpośrednio od liczby krawędzi, zatem gdy było ich mniej prędzej udawało się ustalić odpowiednią kolejność przechodzenia wierzchołków.

Cykl Hamiltona



Złożoność:

- **O(n!) – najgorszy przypadek**

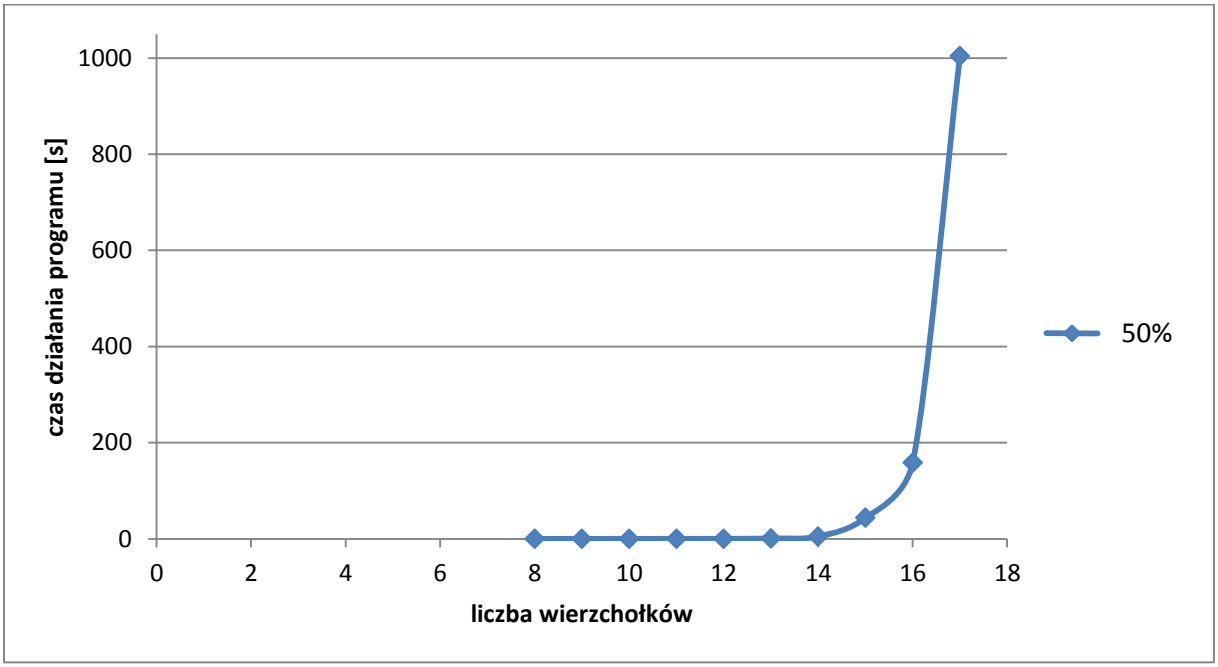
Cykl Hamiltona to taki cykl w grafie, w którym każdy wierzchołek grafu przechodzony jest tylko jeden raz (oprócz pierwszego wierzchołka). W przypadku cyklu Hamiltona nie ustalono, jak dotąd, prostego kryterium, które dla dowolnego grafu pozwoliłoby stwierdzić, czy ów cykl istnieje w danym grafie.

Algorytm wyszukiwania cyklu Hamiltona również opiera się na algorytmie przeszukiwania grafu w głąb. Gdy wywołujemy funkcję rekurencyjną dla danego wierzchołka popychamy go na listę pomocniczą i oznaczamy go jako odwiedzony. Jeżeli na owej liście znajduje już się n wierzchołków (to znaczyłoby, że byliśmy już we wszystkich) sprawdzamy, czy z tego ostatnio dołożonego można wrócić do pierwszego. Jeśli tak, to znaleźliśmy cykl Hamiltona i możemy zakończyć algorytm, zapisując elementy listy pomocniczej na liście wynikowej. Jeżeli na liście pomocniczej nie ma jeszcze n wierzchołków, to wywołujemy funkcję rekurencyjną dla wszystkich nieodwiedzonych do tej pory następników przetwarzanego wierzchołka. Następnie usuwamy ów wierzchołek z kolejki i oznaczamy jako nieodwiedzony. Powoduje to wycofanie się na wyższy poziom rekurencji. Złożoność tego algorytmu jest wykładnicza, ponieważ w najgorszym przypadku dopiero ostatnia permutacja (kolejność ułożenia wierzchołków) okaże się być cyklem Hamiltona (pod warunkiem, że taki cykl w grafie istnieje). Średnio szukanie jednego cyklu Hamiltona w grafie, w którym on istnieje będzie bardziej efektywne, ponieważ rzadko zdarza się, że należy wygenerować wszystkie permutacje. Wyszukiwanie zakończy się po pierwszym odpowiednim ustawieniu wierzchołków. Opłaca się zaimplementować ten algorytm przy pomocy listy następników z tego samego względu, co algorytm znajdowania cyklu Eulera. Tak samo potrzebujemy szybkiego dostępu do wszystkich następników przetwarzanego wierzchołka.

Na wykresie można zauważyć, że algorytm Hamiltona dla gęstości grafu wynoszącej 30% zachowuje się dość nietypowo. Czas działania programu nie rośnie wraz z liczbą wierzchołków, lecz naprzemiennie wzrasta i maleje. Wynika to z faktu, iż trudno określić, jaką średnią złożoność ma ten

algorytm i jak się zachowa. Czasem udaje się wygenerować odpowiednią permutację wierzchołków stosunkowo szybko, a czasem trzeba ich sprawdzić znacznie więcej, wykonując nawroty. Dla większej gęstości grafu algorytm zachowuje się lepiej, ponieważ ilość możliwych permutacji wierzchołków jest podobna, a przez to, iż jest więcej połączeń między nimi znacznie szybciej udaje się znaleźć pierwszy możliwy cykl Hamiltona.

Wyszukiwanie wszystkich cykli Hamiltona w grafie



Złożoność:

- **O(n!)**

Aby znaleźć wszystkie cykle Hamiltona w grafie, po znalezieniu jednego z nich nie przerywaliśmy poszukiwań. Listę pomocniczą zawierającą właśnie znaleziony cykl zapisywaliśmy na listę list wyników i kontynuowaliśmy wyszukiwanie.

Wyszukiwanie wszystkich cykli pokazało, że znalezienie cyklu Hamiltona jest rzeczywiście problemem trudnym. Musieliśmy sprawdzić wszystkie możliwe permutacje i nie mogliśmy (tak jak w poprzednim ćwiczeniu) zatrzymać programu po znalezieniu odpowiedniej. Pokazało to, że jeżeli w grafie nie ma cyklu Hamiltona (czyli szukamy go aż do wyczerpania możliwych permutacji), algorytm znajdowania choćby jednego takiego cyklu zwróci odpowiedź w sensownym czasie tylko dla grafów o liczbie wierzchołków mniejszej od dwudziestu. Z racji tego, że nie znamy kryterium pozwalającego stwierdzić, czy w grafie jest taki cykl, wywołanie wyszukiwania go może trwać bardzo długo. Odpowiedź w rozsądnym czasie, czy taki cykl istnieje, czy nie, również jest możliwa tylko dla grafów o małej liczbie wierzchołków.