Poznań 16 maja 2024

Politechnika Poznańska

Wydział Informatyki i Telekomunikacji

Algorytmy i Struktury Danych, Informatyka (semestr 2)
Sprawozdanie #4 — Algorytmy z powracaniem

wykonał

Konrad Ceglarski

[156912]

1. Wprowadzenie

Algorytmy z powracaniem (*ang. backtracking*) są jedną z podstawowych technik rozwiązywania problemów optymalizacyjnych i kombinatorycznych.

W ramach naszych zajęć laboratoryjnych zaimplementowaliśmy rozwiązania popularnych problemów przy użyciu tej techniki, w tym:

- Poszukiwanie cyklu Hamiltona (algorytm Robertsa-Floresa)
- Poszukiwanie cyklu Eulera (algorytm Fleury'ego)

2. Złożoność obliczeniowa oraz klasy złożoności obliczeniowej wybranych problemów

Cykl Hamiltona		
	Problem przeszukiwania	Problem decyzyjny
Złożoność obliczeniowa	**O(v!)**	**O(v!)**
Klasa złożoności obliczeniowej	**NP-trudny lub silnie NP-trudny**	**NP-zupełny**

`* gdzie v to ilość wierzchołków (vertices)`

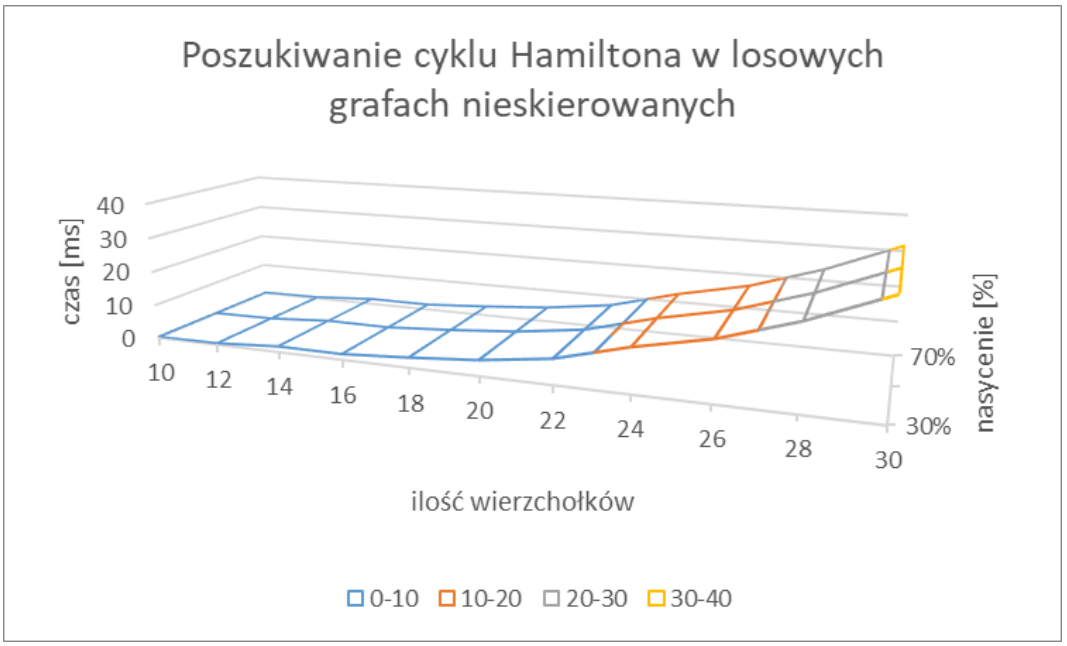
Cykl Eulera		
	Problem przeszukiwania	Problem decyzyjny
Złożoność obliczeniowa	**O(e)**	**O(e)**
Klasa złożoności obliczeniowej	**P**	**P**

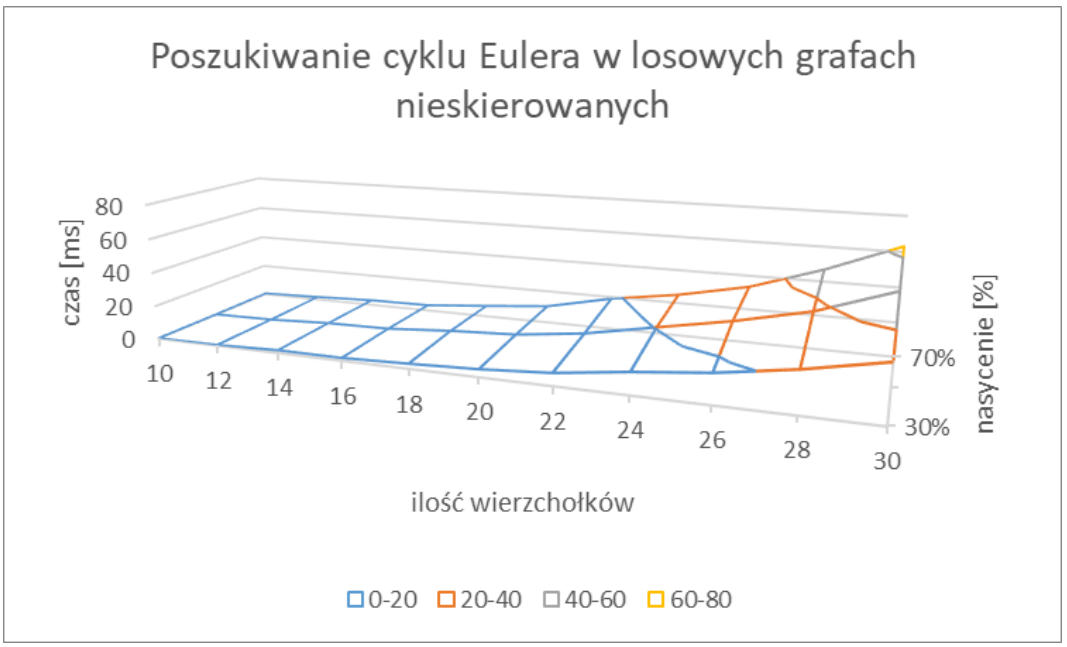
`* gdzie e to ilość krawędzi (edges)`

3. Wykresy obrazujące efektywność wybranych algorytmów

Badany zakres ilości wierzchołków: `od 10 do 30 co 2`

Badany zakres saturacji: `30%, 50% oraz 70%`





`* generowane grafy eulerowskie mają przybliżone nasycenie (z uwagi na sposób generowania), co może wpływać na zakłócenia na wykresie`

4. **Obserwacje związane z działaniem zaimplementowanych algorytmów dla grafów o różnym nasyceniu.**
   Obserwacją, która szczególnie przykuwa uwagę, jest to, że czas rozwiązywania problemów przez te algorytmy jest bardzo czuły i wrażliwy na stopień losowości grafu, co utrudnia wyciągnięcie jednoznacznych wniosków.
   Uogólniając, można jednak stwierdzić, że w najgorszym przypadku oba algorytmy będą potrzebować tym więcej czasu na znalezienie rozwiązania, im bardziej nasycony jest graf oraz im więcej wierzchołków posiada.

5. **Jaką reprezentację maszynową grafu Twoim zdaniem najlepiej zastosować w przypadku każdego algorytmu i dlaczego?**
   Macierz sąsiedztwa w obu przypadkach nie jest złym wyborem, gdyż umożliwia bezpośredni dostęp do sąsiadów każdego wierzchołka. Szczególnie dobrze sprawdza się ona dla grafów gęstych.
   Natomiast dla grafów rzadkich rozsądniejszym rozwiązaniem będzie lista sąsiedztwa (następników lub poprzedników), gdyż zużyje ona mniej pamięci w porównaniu do macierzy sąsiedztwa.

Język implementacji:
Python 🐍 (Python 3.10.10)

Platforma realizacji testów:
Windows 10 x64 (Microsoft Windows [Version 10.0.19045.4170])
CPU: AMD Ryzen 7 3750H (4 cores/8 threads) 2.30 GHz
RAM: 16 GB

Repozytorium:
https://github.com/xKond3i/put_aisd

Źródła:
[1]:cs.put.poznan.pl/mszachniuk/site/teaching/algorytmy-i-struktury-danych
[2]:www.ekursy.put.poznan.pl
[3]:www.eduinf.waw.pl/inf/alg
[4]:pl.wikipedia.org