Zadanie domowe z algorytmiki Seria 2

Wyścig algorytmów dla skojarzeń w grafach dwudzielnych.
Celem tego zadania jest porównanie wydajności znanych algorytmów dla problemu skojarzeń w grafach dwudzielnych. Użyjemy środowiska jupyter notebook, aby zaimplementować w języku python znane algorytmy dla skojarzeń dwudzielnych, porównać ich czasy działania i sporządzić wykresy porównawcze czasów działania poszczególnych algorytmów w zależności od rozmiaru instancji testowych. Aby rozpocząć pracę, należy mieć zainstalowane następujące narzędzia.

Narzędzia. W zadaniu tym użyjemy języka python w połączeniu ze środowiskiem jupyter notebook. Systemy linuksowe mają najczęściej zainstalowany język python3. Instrukcję instalacji środowiska jupyter notebook można znaleźć pod następującym linkiem:

`https://jupyter.org/`

Środowisko jupyter notebook umożliwia wygodny interfejs aby zakodować i uruchamiać pojedyńcze metody, z czego będziemy korzystać. Aby korzystać z narzędzi programowania liniowego, należy doinstalować pakiet python-mip. Instrukcje instalacji oraz użytkowania znaleźć można pod następującym linkiem:

`https://docs.python-mip.com/en/latest/install.html`

Python-mip umożliwia konstruowanie oraz rozwiązywanie programów liniowych w języku python. Domyślnie używa solvera CBC, który jest jednak mało wydajny. Należy zatem zainstalować także osobne narzędzie którym jest solver GUROBI. Instrukcje instalacji i użytkowania można znaleźć pod następującym linkiem:

`https://support.gurobi.com/hc/en-us/articles/14799677517585-Getting-Started-with-Gurobi-Optimizer`

Kuszące wydaje się skorzystanie z możliwości instalacji GUROBI wyłącznie dla pythona za pomocą komendy

```
python -m pip install gurobipy
```

jednak umożliwi to rozwiązywanie wyłącznie małych programów liniowych. Dużo lepszym rozwiązaniem jest pełna instalacja na licencji akademickiej, a następnie użycie GUROBI z poziomu python-mip.

Treść zadania. Głównym celem zadania będzie zaimplementowanie kilku metod, z których każda będzie implementowała algorytm dla znajdowania naliczniejszego skojarzenia w grafie dwudzielnym. Każda taka metoda powinna odczytywać graf z pliku, dokonać obliczeń, a następnie zwrócić wynik. Czas obliczeń każdej metody może zostać zmierzony przy użyciu biblioteki time.

a) Zaimplementuj metodę HK, która implementuje algorytm Hopcrofta-Karpa. Algorytm Hopcrofta-Karpa znajduje skojarzenie w grafach dwudzielnych w czasie $O(|E| \cdot \sqrt{|V|})$, zaś jego opis można znaleźć np. tu:

   `https://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm`

   Twoja implementacja powinna być efektywna, t.j. faktycznie działać w czasie $O(|E|\sqrt{|V|})$.

b) Korzystając z biblioteki python-mip, zaimplementuj metodę, która rozwiązuje problem skojarzeń dwudzielnych za pomocą programu liniowego. Będziemy testować dwa warianty tej metody, różniące się wyłącznie zastosowanym solverem. Niech metoda LPcbc oznacza wariant używający domyślny solver CBC, natomiast jako LPgur oznaczymy wariant używający solvera GUROBI.

c) Porównaj działanie metod HK, LPcbc oraz LPgur na grafach losowych 300×300 wierzchołków przy prawdopodobieństwie krawędzi rosnącym od 0 do 1. Narysuj stosowny wykres używając biblioteki pyplot.

d) Dla dowolnego n podaj przykład grafu o $2n$ wierzchołkach, dla którego algorytm Hopcrofta-Karpa wykona $\Omega(\sqrt{n})$ faz, czyli zadziała w czasie $\Omega(|E|\sqrt{|V|})$.

e) Porównaj czasy działania metod HK, LPcbc i LPgur na instancjach trudnych dla algorytmu Hopcrofta-Karpa. Narysuj stosowny wykres czasów działania przy rosnącym rozmiarze instancji n, kończąc na n rzędu 500000.

f) Zaproponuj jak najlepszą heurystykę która na tych instancjach działa wydajniej niż HK. Zaimplementuj ją nazywając stosowną metodę AlternativeToHK. Czy potrafisz znaleźć trudny przykład dla Twojej heurystyki ?

f*) Alternatywnie, metoda AlternativeToHK może implementować algorytm rangowy z pracy 'Online bipartite matching in offline time', który również znajduje skojarzenie w grafie dwudzielnym w czasie $O(|E|\sqrt{|V|})$, ale w zupełnie inny sposób niż algorytm Hopcrofta-Karpa. Algorytm rangowy jest ciekawy i prosty w implementacji. Pracę 'Online bipartite matching in offline time', gdzie można go znaleźć, udostępnimy na platformie moodle. Współautorka tej pracy chętnie wytłumaczy ten algorytm chętnym w ramach konsultacji.

g) Przetestuj metody HK, LPcbc, LPgur i AlternativeToHK na instancjach własnego pomysłu. Narysuj stosowne wykresy porównawcze. Czy na podstawie testów możesz wywnioskować coś na temat asymptotycznych czasów działania zastosowanych solverów?

Rozwiązania. Implementacje wszystkich wymaganych metod, generatorki grafów oraz wykresy powinny znaleźć się w pliku jupyter notebook o rozszerzeniu .ipynb, który należy wysłać na moodla. Dodatkowo, wymagany jest raport (najlepiej w pliku .pdf) podsumowujący porównanie wymaganych metod, zawierający stosowne wykresy, opisy instancji testowych oraz wnioski. Rozwiązania należy nadsyłać w terminie do piątku 2024-04-26 godz. 20:00. Całość zadania bez algorytmu rangowego warta jest 20 punktów. Poprawna implementacja algorytmu rangowego i porównanie go z innymi da bonusowe 5 punktów