Dominik Wawszczak
numer indeksu: 440014
numer grupy: 1

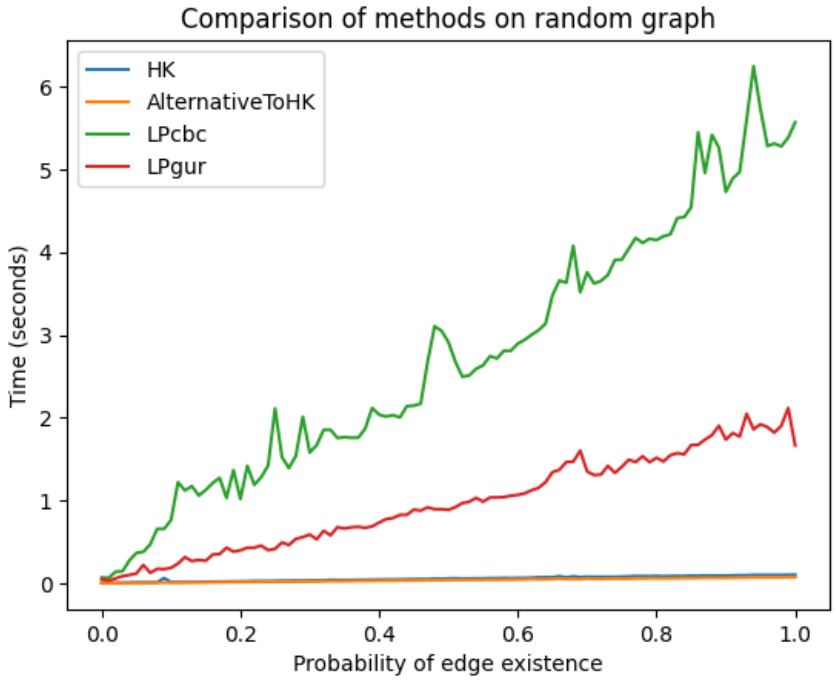
Algorytmika
praca domowa 2

Wszystkie porównania były uruchamiane na laptopie z procesorem Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, stąd długie czasy działania. Na komputerze stacjonarnym ze znacznie mocniejszym procesorem niestety nie udało mi się aktywować licencji na GUROBI w WSLu.

a) Metodę `HK` będącą implementacją algorytmu Hopcrofta-Karpa można znaleźć w pliku `dw440014-assignment2.ipynb`.

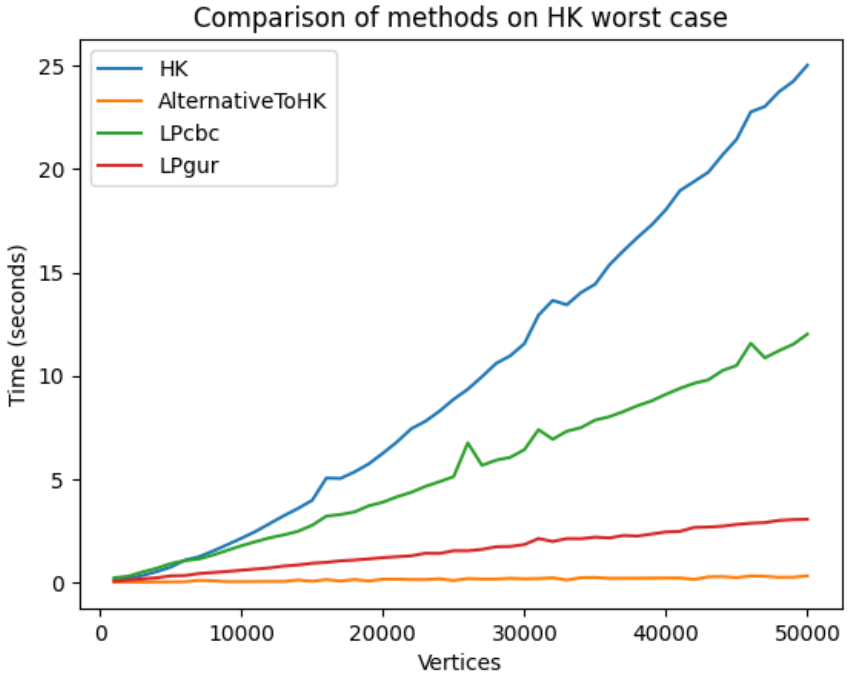
b) Metody `LPcbc` oraz `LPgur` również znajdują się w tym pliku.

c) Poniżej znajduje się wykres będący porównaniem działania metod na gęstych losowych grafach o $n = 300$ wierzchołkach.



d) Przykładem takiego grafu jest k rozłącznych ścieżek o długościach kolejno $1, 3, \ldots, 2k-1$, dla największego k spełniającego $\binom{k+1}{2} \leqslant n$, czyli dla k rzędu $\sqrt{n}$. Jeżeli ponumerujemy wierzchołki w odpowiedniej kolejności, to po pierwszej iteracji algorytmu każda ze wspomnianych ścieżek będzie ścieżką powiększającą. Ścieżki te mają parami różne długości, zatem każda z nich zostanie znaleziona w innej iteracji, skąd wnioskujemy, że algorytm wykona ich $O(\sqrt{n})$.

e) Ze względu na długi czas działania algorytmu na instancjach z podpunktu d), zdecydowałem się przetestować metody dla n do 50 000. Wyniki prezentują się następująco:



f) Zaproponowaną przeze mnie alternatywą, której implementację można znaleźć w metodzie `AlternativeToHK`, jest algorytm znany pod nazwą *Turbo matching*, głównie wśród uczestników konkursów typu OI czy ICPC. Nie znam jego złożoności, ale jest bardzo szybki w praktyce i prosty w implementacji – stąd jego popularność na tego typu konkursach. Nie znam również trudnego przykładu dla tego algorytmu, choć słyszy się legendy o pewnych „rosyjskich testach", na których podobno działa bardzo wolno.

g) Dodatkowo przetestowałem wszystkie metody na grafach rzadkich (każdemu wierzchołkowi losujemy sąsiada). Poniżej znajduje się rezultat porównania.