Музичук Теодор, Серивко Остап

Дискретна математика, 2 семестр

Лабораторна робота 1, завдання 1

20 лютого 2022

Порівняння ефективності алгоритмів Краскала та Прима

Двома найпопулярнішими алгоритмами для знаходження мінімального каркасу у зваженому графі є алгоритми Прима та Краскала. Ця задача є релевантною у багатьох галузях, і у різних випадках може бути доцільним використання того чи іншого алгоритму. Мета дослідження полягає в тому, щоб визначити, для яких графів ефективніший один алгоритм, а для яких - інший.

ЕКСПЕРИМЕНТ

Були написані власні імплементації обох алгоритмів мовою програмування Python, еквівалентні за складністю до вбудованих у бібліотеку NetworkX. Далі за допомогою бібліотеки matplotlib були реалізовані візуалізації роботи алгоритмів на графах з малою кількістю вершин для наочності (рисунки 1-4). Після цього, кожен алгоритм був протестований на швидкодію на графах із кількістю вершин від 1 до 991 (крок 10) та значеннями заповненості 0.2, 0.5 та 1 (граф K_n) (рисунки 5-7). Для порівняння на графіках також є оцінка швидкодії вбудованого алгоритму NetworkX.

ПРОГРАМНИЙ КОД ЕКСПЕРИМЕНТУ

Весь код, включно з кодом алгоритмів та візуалізації, є у відкритому репозиторії проекту на GitHub: redn1nja/Discrete_lab1.

Модулі:

1. prim.py - реалізація алгоритму Прима. Код алгоритму знаходиться у функції generate_mst_prim.
2. kruskal.py - реалізація алгоритму Краскала. Код алгоритму знаходиться у функції generate_mst_kruskal.
3. graph_gen.py - модуль із функцією для генерації графів із GraphGeneration.ipynb
4. visualize.py - модуль із функціями для візуалізації роботи алгоритмів. Візуалізація робилася через pyplot. Функція visualize_mstp зображує переданий їй граф та дерева, згенеровані трьома алгоритмами: Прима, Краскала та вбудованим. Функція plot_algorithm_comparisons зображує порівняльний графік швидкодії трьох алгоритмів для заданих проміжку кількостей вершин та завершеності графа.

СПЕЦИФІКАЦІЯ КОМП’ЮТЕРА (Рис.8)

ОС: Gentoo/Linux x86_64 (ядро Linux 5.15.23)

Версія Python: 3.9.9

Процесор: Intel i5-8350U (4 ядра, 8 логічних)

Тактова частота: 3.600Ггц

Оперативна пам’ять: 16 Гб

АНАЛІЗ ОТРИМАНИХ ДАНИХ

Обчислювальна складність алгоритмів Краскала та Прима в ідеальній імплементації - O(nlog(n)), де n - кількість вершин або ребер (еквівалентні для оцінки складності). Щоправда, бачимо з графіків, що при малій кількості ребер алгоритм Краскала дещо менш ефективний ніж алгоритм Прима, але при збільшенні цієї кількості алгоритми стають рівними, а при завершеності 1 Краскал навіть дещо швидший. Звідси можемо зробити висновок, що для повних графів краще використовувати алгоритм Краскала, але різниця несуттєва, а для графів де ребер порівняно мало Прим суттєво швидший, хоча навіть при кількості вершин більше 900 різниця лише в секунду.

ВИСНОВОК

При меншій кількості ребер у графі алгоритм Прима має більшу ефективність, а алгоритм Краскала перемагає у швидкодії коли ребер більше. При великій кількості вершин та завершеності графа близькій одиниці, Краскала використовувати доцільніше.

Рис.1: Зважений граф, згенерований функцією *gnp_random_connected_graph*

(8 вершин, 18 ребер, завершеність 0.5, сумарна вага: 78)

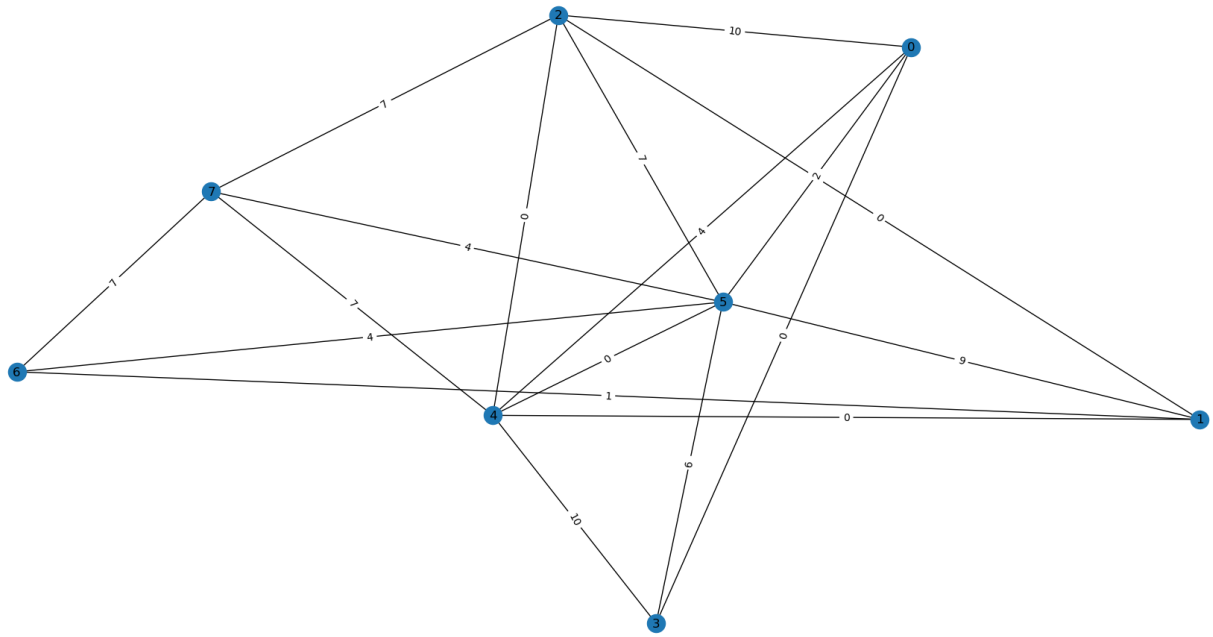

Рис.2: Дерево, побудоване алгоритмом Прима (вага: 7).

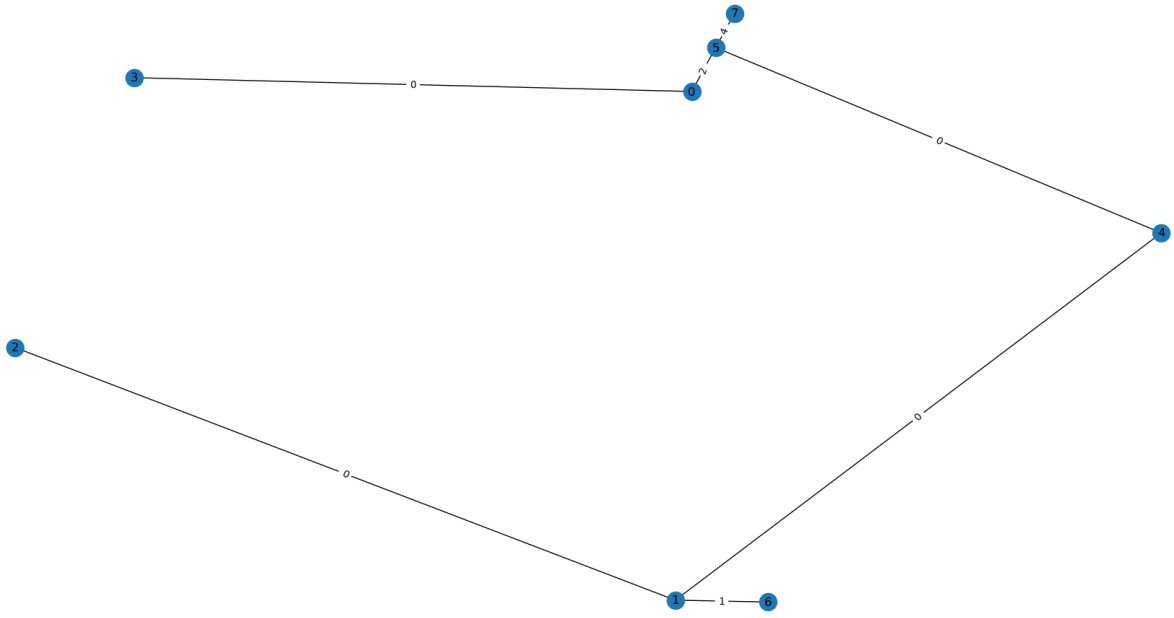

Рис.3: Дерево, побудоване алгоритмом Краскала (вага: 7).

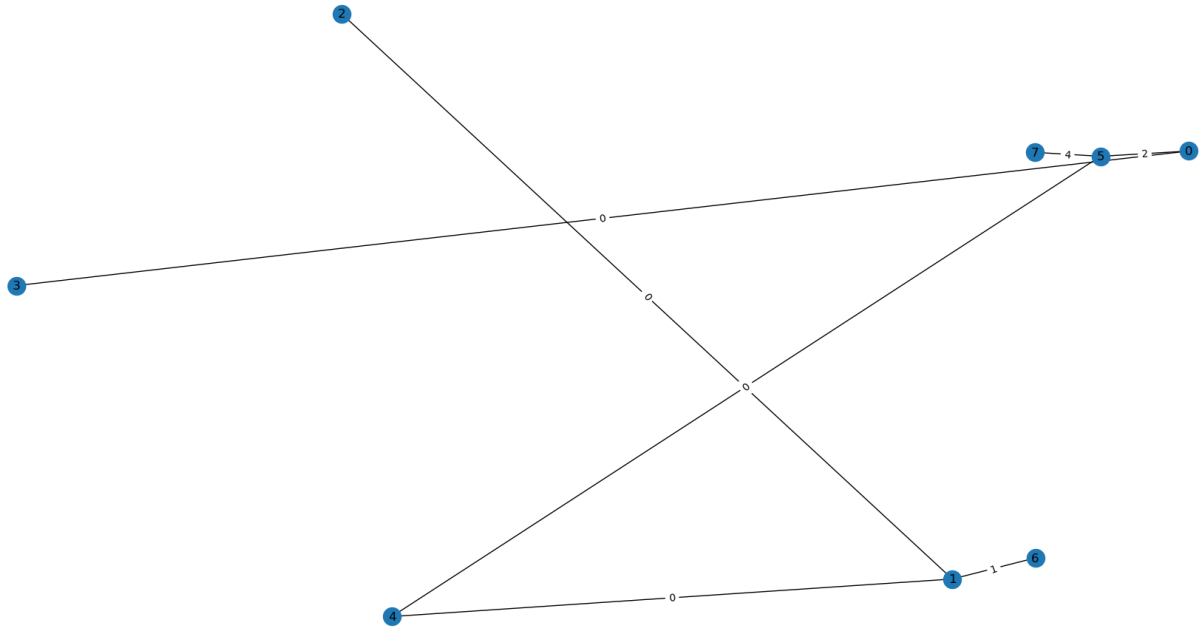

Рис. 4: Дерево, побудоване вбудованим алгоритмом NetworkX (вага: 7).

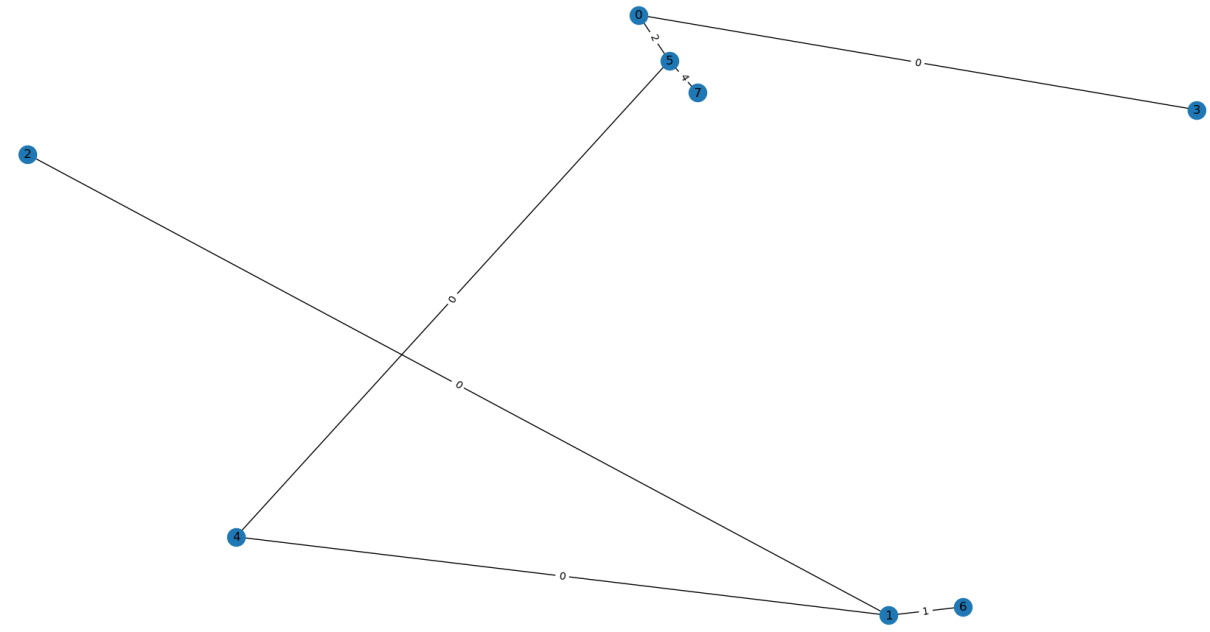

Рис.5: Швидкодія алгоритмів на графах завершеністю 0.2.

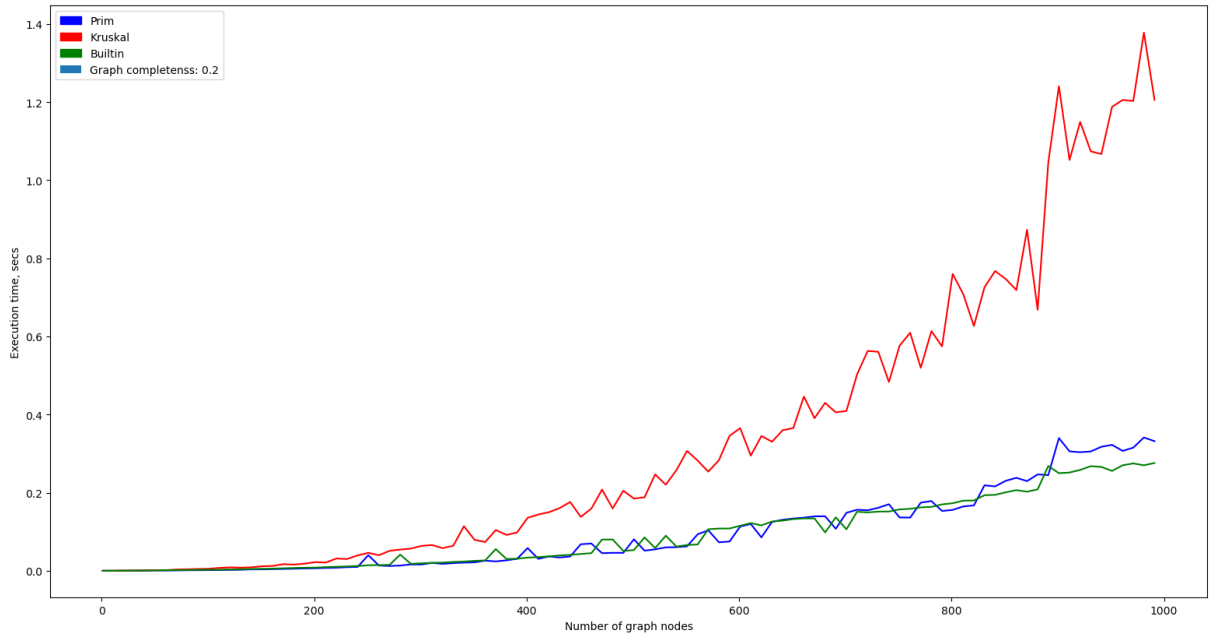

Рис.6: Швидкодія алгоритмів на графах із завершеністю 0.5.

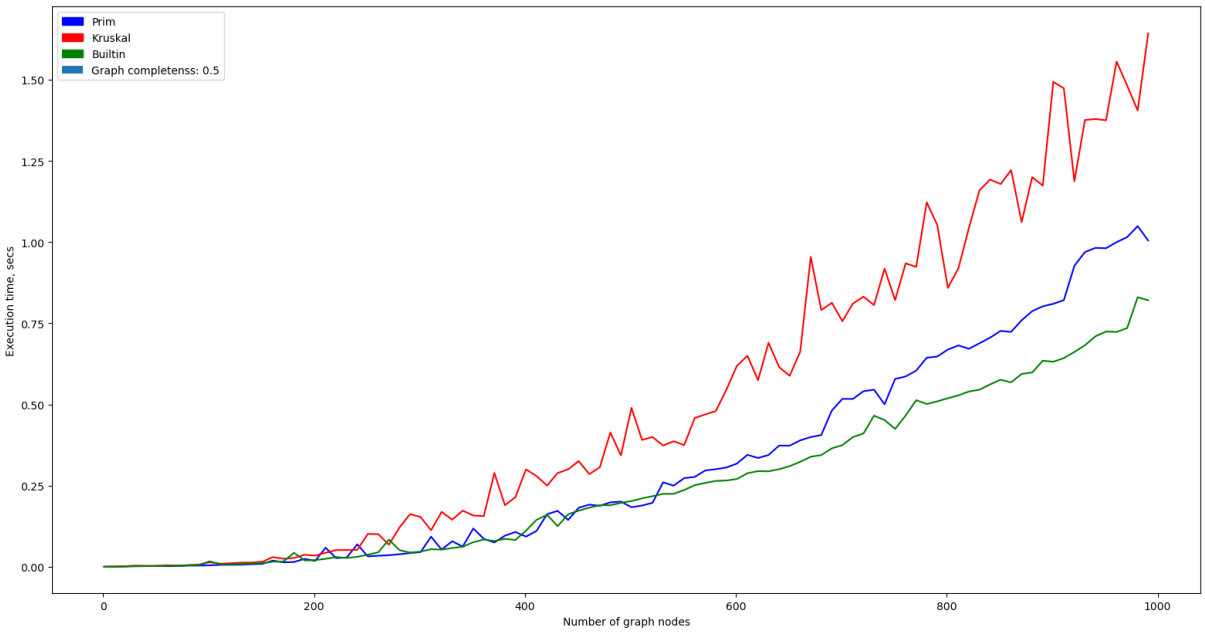

Рис.7: Швидкодія алгоритмів на графах K_n (завершеність 1).

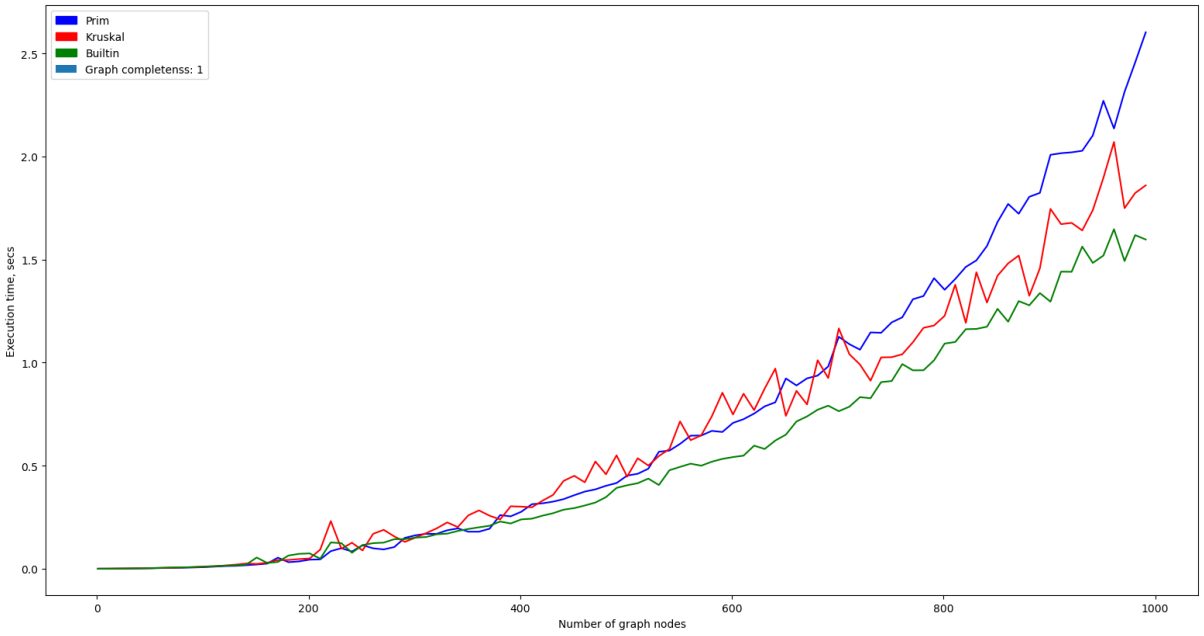

Рис.8: Вихідні дані команди neofetch

```
┌─[~/Documents/Discr/Lab1/Discrete_lab1]─[vepbxer@vepbxerpc]─[0]─[616]
└─[:)] % neofetch
          -/oyddmdhs+:.                vepbxer@vepbxerpc
      -odNMMMMMMMMNNNmhy+-`            -----------------
    -yNMMMMMMMMMMMNNNmmdhy+-           OS: Gentoo/Linux x86_64
  `omMMMMMMMMMMMMNmdmmmmddhhy/`       Host: Latitude 5590
  omMMMMMMMMMMMNhhyyyohmdddhhhdo`      Kernel: 5.15.23-gentoo
.ydMMMMMMMMMMdhs++so/smdddhhhhdm+`     Uptime: 1 day, 20 hours, 45 mins
 oyhdmNMMMMMMMNdyooydmddddhhhhyhNd.    Packages: 1253 (emerge), 6 (flatpak)
  :oyhhdNNMMMMMMMNNNmmdddhhhhhyymMh    Shell: zsh 5.8
    .:+sydNMMMMMNNNmmmdddhhhhhhmMmy    Resolution: 1920x1080
       /mMMMMMNNNmmmdddhhhhhmMNhs:     WM: sway
    `oNMMMMMMMNNNmmmddddhhdmMNhs+`     Theme: Adwaita-dark [GTK2/3]
  `sNMMMMMMMNNNmmmdddddmNMmhs/.        Icons: Windows XP SP3 [GTK2/3]
 /NMMMMMMMNNNNmmmdddmNMNdso:`          Terminal: alacritty
+MMMMMMNNNNNmmmmdmNMNdso/-             CPU: Intel i5-8350U (8) @ 3.600GHz
yMNNNNNNNmmmmmNNMmhs+/-`               GPU: Intel UHD Graphics 620
/hMNNNNNNNNMNdhs++/-`                  Memory: 3170MiB / 15866MiB
`/ohdmmddhys+++/:.`
  `-//////:--.
```