Antroji užduotis

Tikslas: įsisavinti rūšiavimo algoritmus skirtingiems objektams.

1. Modifikuoti "heapsort" rūšiavimą: parašyti programą, kuri rūšiuoja labai didelius kintamo ilgio sveikus skaičius.
2. Modifikuoti "heapsort" rūšiavimą 3-ainiams "heap", kai kiekviena viršūnė turi tris sūnus: parašyti efektyvią rūšiavimo programą.
3. Parašyti efektyvią programą, kuri realizuoja įterpimo (insert) operaciją *d*-ainiams "heap", ištirti koks yra jos sudėtingumas *d* ir elementų skaičiaus *n* atžvilgiu.
4. Parašyti efektyvią programą, kuri realizuoja didžiausio elemento išmetimo operaciją (delete max) ir atstato heap struktūrą *d*-ainiams heap, ištirti koks yra jos sudėtingumas *d* ir elementų skaičiaus *n* atžvilgiu.
5. Turime penkias aibes skaičių, išrūšiuotų kiekvienoje aibėje didėjimo tvarka. Realizuoti suliejimo operaciją heap struktūros pagalba.
6. Realizuoti operacijas su eilute (paprasta eilute, ne prioritetine) panaudojant heap struktūrą, kai eilutės elementai yra vieno kintamojo daugianariai.
7. Realizuoti operacijas su steku, panaudojant heap struktūrą, kai steko elementai yra labai dideli kintamo ilgio sveiki skaičiai.
8. Modifikuoti rūšiavimą Shellsort metodu: parašyti programą, kuri rūšiuoja vieno kintamojo daugianarius.
9. Modifikuoti radix-exchange rūšiavimą: parašyti programą, kuri rūšiuoja labai didelius kintamo ilgio sveikus skaičius.
10. Modifikuoti radix-straight rūšiavimą: parašyti programą, kuri rūšiuoja vieno kintamojo daugianarius.
11. Sukurti algoritmą ir parašyti programą, kuri duotai skirtingų skaičių aibei *S* suranda *k* skaičių, artimiausių aibės medianai. Algoritmo sudėtingumas – *O(n)*.
12. Sukurti *O (log n)* sudėtingumo algoritmą ir parašyti programą, kuri dviems surūšiuotų skaičių vektoriams, kiekvienas po *n* skaičių, suranda medianą visiems *2n* skaičių iš abiejų vektorių.
13. Turime *n* įrašų, kurių raktinės reikšmės yra tarp *1* ir *k*. Modifikuoti pasiskirstymų skaičiavimą (counting sort) taip, kad įrašai gali būti rūšiuojami nenaudojant papildomos atminties (sorted in place), o algoritmo sudėtingumas būtų *O(n+k)*.
14. Sukurti efektyvų algoritmą ir parašyti programą, kuri apjungia dvi prioritetines eilutes į vieną (operacija *join*), eilutės saugomos heap struktūrose.
15. Sukurti efektyvų algoritmą ir parašyti programą, kuri apjungia du netiesioginius heap į vieną (operacija *join*).
16. Modifikuoti greito rūšiavimo (quicksort) algoritmą ir parašyti programą, kad šis rūšiavimas būtų stabilus tiesiniams sąrašams.
17. Parašyti programą, kuri surastų failą greito rūšiavimo (quicksort) algoritmo geriausiam atvejui (kiekviena particija dalina failą į du failus, kurių ilgiai skiriasi ne daugiau kaip 1).
18. Realizuoti rekursyvų greito rūšiavimo algoritmą, empiriškai nustatant reikalingą steko dydį, kai rūšiuojama 1000, 10000, 100000, 1000000 kintamo ilgio žodžių.
19. Parašyti programą įterpimo rūšiavimo algoritmui, kuri dirba daug kartų ir rūšiuoja įvairių dydžių failus, matuodama darbo laiką, skaičiuodama lyginimo operacijų kiekį, ir sudaro tokių darbo laikų ir kiekių pasiskirstymą.
20. Parašyti programą išrinkimo rūšiavimo algoritmui, kuri dirba daug kartų ir rūšiuoja įvairių dydžių failus, matuodama darbo laiką, skaičiuodama lyginimo operacijų kiekį, ir sudaro tokių darbo laikų ir kiekių pasiskirstymą.

21. Parašyti programą burbuliuko rūšiavimo algoritmui, kuri dirba daug kartų ir rūšiuoja įvairių dydžių failus, matuodama darbo laiką ir sudaro tokių darbo laikų pasiskirstymą.
22. Parašyti programą shellsort rūšiavimo algoritmui, kuri dirba daug kartų ir rūšiuoja įvairių dydžių failus, matuodama darbo laiką ir sudaro tokių darbo laikų pasiskirstymą (naudojant tą pačią inkrementinę seką).
23. Parašyti programą greito rūšiavimo (quicksort) algoritmui, kuri dirba daug kartų ir rūšiuoja įvairių dydžių failus, matuodama darbo laiką, skaičiuodama lyginimo operacijų kiekį, ir sudaro tokių darbo laikų ir kiekių pasiskirstymą.
24. Parašyti programą, kuri rūšiuoja duomenis pasiskirstymų skaičiavimo metodu, kai duomenys yra ilgi skaičiai, o rūšiavimo raktu imami skaitmenys, esantys fiksuotose pozicijose (pozicijų intervale).
25. Parašyti programą, kuri rūšiuoja N atsitiktinių skaičių greito rūšiavimo metodu ir nustato vidurinį kiekį particijų, turinčių 0, 1, 2 arba 3 elementus.
26. Sukurti efektyvų algoritmą ir parašyti programą, kuri iš N duotų labai didelių skaičių surastų k mažiausių, kai skaičiai lyginami pagal jų reikšmes tam tikrame pozicijų intervale.
27. Parašyti programas *heapsort*, kada naudojamos dvi skirtingos operacijos: išmesti mažiausią ir išmesti didžiausią. Palyginti jų darbą (operacijų pasiskirstymams) esant tai pačiai duomenų sekai.
28. Modifikuoti rūšiavimą shellsort metodu: parašyti programą, kuri rūšiuoja abėcėlės tvarka kintamo ilgio labai ilgus žodžius.
29. Modifikuoti rūšiavimą pasiskirstymų skaičiavimo įterpimo metodu: parašyti programą, kuri rūšiuoja abėcėlės tvarka kintamo ilgio laba ilgus žodžius.
30. Modifikuoti rūšiavimą heapsort metodu: parašyti programą, kuri rūšiuoja abėcėlės tvarka kintamo ilgio labai ilgus žodžius.