

3.4 Powiemy, że dwa słowa są podobne wtedy i tylko wtedy, gdy zawierają jednakowe liczby wystąpień tych samych znaków. Danych jest n słów nad alfabetem m -znakowym $\{1, 2, \dots, m\}$. Zaproponuj algorytm, który stwierdza, ile jest wśród nich różnych klas słów podobnych. Twój algorytm powinien działać w czasie $O(R + m)$, gdzie R jest sumą długości wszystkich słów.

4.1 W tym zadaniu analizujemy algorytm QuickSort, w którym za element dzielący wybiera się medianę z $(a[l], a[\lfloor (l + r)/2 \rfloor + 1], a[r])$ (zobacz wykład 3). Rozważmy permutacje

[illegible]

4.2 Niech n będzie liczbą całkowitą większą od 1, a

$$X = \{(x, y) : x = 0, 1, \dots, n-1, y = 0, 1, 2, 3, 4\}$$

będzie zbiorem punktów na płaszczyźnie. Danych jest n różnych prostych, z których każda przechodzi przez dwa różne punkty ze zbioru X , różniące się zawsze pierwszą współrzędną. Każda prosta zadana jest przez parę punktów $\{(x_1, y_1), (x_2, y_2)\}$, gdzie $x_1 < x_2$. Zaprojektuj algorytm, który w czasie liniowym posortuje wszystkie proste niemalejąco względem ich kątów nachylenia do osi OX .

4.4 Dla dodatniej liczby całkowitej d , $d > 1$, d -kopcem typu MIN, nazywamy zupełne drzewo d -arne z kluczami rozmieszczonymi w porządku kopcowym typu MIN. Zaproponuj wydajną implementację d -kopca w tablicy i dokonaj analizy złożoności obliczeniowych operacji kolejki priorytetowej: Ini, Min, DeleteMin, Insert i DecreaseKey. W jaki sposób dobrać d , żeby dostać jak najszybszą implementację algorytmu Dijkstry, uwzględniającą liczbę krawędzi w grafie.

Do samodzielnej pracy

- 4.3 Podaj permutację liczb $1, 2, \dots, 7$, dla której algorytm HeapSort (w wersji z wykładu) wykona największą liczbę porównań.

Uwaga: należy wziąć pod uwagę obie fazy algorytmu budowę kopca i właściwe sortowanie.

- 4.5 Drzewem lewicowym nazywamy drzewo binarne, w którym dla każdego wierzchołka długość skrajnie prawej ścieżki w jego lewym poddrzewie jest nie mniejsza od długości skrajnie prawej ścieżki w jego prawym poddrzewie.

Kopcem lewicowym typu Min nazywamy drzewo lewicowe z kluczami rozmieszczonym w porządku kopcowym typu Min. Zaproponuj implementację kopca lewicowego umożliwiającą wydajne wykonywanie operacji kolejki priorytetowej: Ini, Min, DeleteMin, Insert, DecraseKey.

Uwaga: zacznij od implementacji operacji $\text{Join}(L_1, L_2)$ łączenia dwóch kopców lewicowych w jeden kopiec.