Zadanie 1 [5 punktów]

Zaprojektuj optymalny algorytm pod względem pesymistycznej liczby porównań, który znajduje dwa środkowe elementy w zbiorze czterech elementów. Dowiedź poprawności swojego rozwiązania.

*Rozwiązanie:*

*Dolna granica: # możliwych wyników wynosi 12; zatem w pesymistycznym przypadku potrzeba 4 porównań.*

*Górna granica: porównujemy w parach, a następnie odrzucamy minimum z mniejszych elementów w parach i maksimum z większych elementów w parach.*

Zadanie 2 [10 punktów]

**Drzewem klasówkowym** nazywamy pełne drzewo binarne, w którym klucze są rozmieszczone zgodnie z następującą regułą: *dla każdego węzła x najmniejszy klucz w poddrzewie o korzeniu x znajduje się w jego lewym podrzewie.*

Zaproponuj implementację drzewa klasówkowego w sposób, który umożliwia wydajne wykonywanie operacji kolejki priorytetowej:

Ini:: mając dane *n*=2^*k*-1 kluczy zbuduj *n*-węzłowe drzewo klasówkowe

Min:: podaj wartość najmniejszego klucza w drzewie

ChangeKey(x,k):: zmień wartość klucza we wskazanym węźle *x* na *k*

Uzasadnij poprawność swoich rozwiązań oraz dokonaj analizy ich złożoności obliczeniowej.

*Rozwiązanie:*

*Każdy węzeł w drzewie jest rekordem, w którym trzymamy:*

*wskaźnik do lewego syna, wskaźnik do prawego syna, wskaźnik do ojca, wskaźnik do skrajnego lewego potomka, klucz.*

*Drzewo jest dostępne przez korzeń.*

*Min: element najmniejszy jest w skrajnym lewym potomku korzenia; czas O(1).*

*Change(x,k):*

*Umieszczamy k w x.*

1. *Wartość klucza zmalała: jeśli nowy klucz nie jest mniejszy od dotychczas najmniejszego w poddrzewie, to koniec. W przeciwnym razie zamieniamy klucz k z najmniejszym kluczem w podrzewie o korzeniu x. Następnie idziemy z x w górę drzewa i zamieniamy lewego syna z prawym o ile w prawym podrzewie najmniejszy klucz jest mniejszy od najmniejszego klucza w lewym poddrzewie.*
2. *Wartość klucza wzrosła: gdy x nie jest liściem, to koniec. Gdy x jest liściem , poprawiamy drzewo tak jak w pukcie 1).*

*Czas O(log n).*

*Ini: umieszczamy klucze w węzłach drzewa i budujemy je od dołu, zamieniając w razie potrzeby synów węzłów z kolejnych poziomów oraz dodatkowo podmieniając klucz w skrajnie lewym potomku z kluczem w korzeniu każdego poddrzewa, jeśli jest taka potrzeba.*

*Czas O(n).*

Zadanie 3 [5 punktów]

Danych jest *k* uporządkowanych list o długościach będących parami różnymi potęgami dwójki. Zaproponuj wydajny algorytm scalenia tych list w jedną listę uporządkowaną. Uzasadnij poprawność swojego algorytmu i dokonaj analizy jego złożoności obliczeniowej ze względu na liczbę porównań wykonywanych podczas scalania.

*Rozwiązanie: scalamy zawsze dwie najkrótsze listy. Łączna liczba porównań, to co najwyżej dwa razy suma długości wszystkich list. Żeby to udowodnić wystarczy każdej liści przypisać 2 razy tyle żetonów ile wynosi jej długość. 1 żeton opłaca 1 porównanie.*