Řazení II
cvičení

1.

- Merge sort

a) lze napsat tak, aby nebyl stabilní
b) je stabilní, protože jeho složitost je $\Theta(n*\log(n))$
c) je nestabilní, protože jeho složitost je $\Theta(n*\log(n))$
d) je rychlý ($\Theta(n*\log(n))$) právě proto, že je stabilní
e) neplatí o něm ani jedno předchozí tvrzení

2a.

- V určitém problému je velikost zpracovávaného pole s daty rovna rovna $2n-2$, kde n charakterizuje velikost problému. Pole se řadí pomocí Merge Sort-u.
- S použitím $\Theta/O/\Omega$ symboliky vyjádřete asymptotickou složitost tohoto algoritmu nad uvedeným polem v závislosti na hodnotě n. Výsledný vztah předložte v co nejjednodušší podobě.

2b.

- V určitém problému je velikost zpracovávaného pole s daty rovna rovna $2n^3$,kde n charakterizuje velikost problému. Pole se řadí pomocí Merge sort-u.
- S použitím $\Theta/O/\Omega$ symboliky vyjádřete asymptotickou složitost tohoto algoritmu nad uvedeným polem v závislosti na hodnotě n. Výsledný vztah předložte v co nejjednodušší podobě.

3.

- Merge sort řadí pole, které obsahuje $n^3 + \log(n)$ položek. Asymptotická složitost tohoto řazení je

a) $\Theta(n^3 + \log(n))$
b) $\Theta(n^2 \cdot \log(n^3))$
c) $\Theta(n^3 \cdot \log(n))$
d) $\Theta(n \cdot \log(n))$
e) $\Theta(n^3 + n^2)$

4a.

- Merge sort řadí pole šesti čísel: { 8, 1, 7, 6, 4, 2 }.

Situace v aktuálně řazeném poli bude těsně před provedením posledního slévání (merging) následující:

a) 1 7 8 2 4 6
b) 1 2 4 6 7 8
c) 8 7 6 4 2 1
d) 1 2 4 7 8 6
e) 2 4 6 1 7 8

4b.

Merge sort řadí pole šesti čísel: { 9, 5, 8, 7, 2, 0 }.

Situace v aktuálně řazeném poli bude těsně před provedením posledního slévání (merging) následující:

a) 0 2 7 5 8 9
b) 0 2 5 7 8 9
c) 2 5 7 0 8 9
d) 7 8 9 0 2 5
e) 5 8 9 0 2 7

5.

- Předložíme-li Heap sort-u vzestupně seřazenou posloupnost délky n, bude složitost celého řazení

a) $\Theta(n)$, protože Heap sort vytvoří haldu v čase $\Theta(n)$
b) $\Theta(n^2)$, protože Heap sort vytvoří haldu v čase $\Theta(n^2)$
c) $\Theta(n \cdot \log_2(n))$, protože je to složitost vytvoření haldy
d) $\Theta(n \cdot \log_2(n))$, protože je to složitost zpracování haldy
e) $\Theta(n)$, protože vytvoření i zpracování haldy bude mít právě tuto složitost

6.

- Heap sort

a) není stabilní, protože halda (=heap) nemusí být pravidelným stromem
b) není stabilní, protože žádný rychlý algoritmus ($\Theta(n*\log(n))$) nemůže být stabilní
c) je stabilní, protože halda (=heap) je vyvážený strom
d) je stabilní, protože to zaručuje pravidlo haldy
e) neplatí o něm ani jedno předchozí tvrzení

7a.

- Která z následující posloupností představuje haldu o čtyřech prvcích uloženou v poli?

a) 1 3 4 2
b) 1 4 2 3
c) 1 2 4 3
d) 2 3 4 1

7b.

- Která z následující posloupností představuje haldu o čtyřech prvcích uloženou v poli?

a) 1 3 4 2
b) 1 3 2 4
c) 1 4 2 3
d) 2 3 1 4

8a.

- Je dána halda uložená v poli. Proveďte první krok fáze řazení v Heap Sortu (řazení haldou), tj.

a) zařaďte nejmenší prvek haldy na jeho definitivní místo v seřazené posloupnosti a

b) opravte zbytek tak, aby opět tvořil haldu.

1 5 2 17 13 24 9 19 23 22

8b.

- Je dána halda uložená v poli. Proveďte první krok fáze řazení v Heap Sortu (řazení haldou), tj.

a) zařaďte nejmenší prvek haldy na jeho definitivní místo v seřazené posloupnosti a

b) opravte zbytek tak, aby opět tvořil haldu.

4 8 11 12 9 18 13 17 21 25

9.

- Zadanou posloupnost v poli seřaďte ručně pomocí heap sortu. Registrujte stav v poli po každém kroku. Nejprve po každé opravě částečné haldy od prostředku pole směrem k začátku, tj. při první fázi. Potom také po každé opravě haldy a přenesení prvku z vrcholu haldy na za konec haldy.

23 29 27 4 28 17 1 24 6 30 19

10.

- Vysvětlete, jak je nutno modifikovat Heap sort, aby po jeho skončení pole obsahovalo prvky seřazené vzestupně. Algoritmus musí být stejně rychlý, nejen asymptoticky, a nesmí používat žádné další datové struktury nebo proměnné.

11.

- Následující dvojici řazení je možno implementovat tak, aby byla stabilní

a) Heap sort a Insertion sort
b) Selection sort a Quick sort
c) Insertion sort a Merge sort
d) Heap sort a Merge sort
e) Radix sort a Quick sort

12.

- Poskytneme-li Quick Sort-u (Q) a Merge sort-u (M) stejná data k seřazení, platí

a) Q bude vždy asymptoticky rychlejší než M
b) M bude vždy asymptoticky rychlejší než Q
c) někdy může být Q asymptoticky rychlejší než M
d) někdy může být M asymptoticky rychlejší než Q
e) oQ i M budou vždy asymptoticky stejně rychlé

13.

- Implementujte nerekurzivní variantu Merge sortu s využitím vlastního zásobníku. Vaše implementace nemusí usilovat o maximální rychlost, stačí, když dodrží asymptotickou složitost Merge sortu. Jednotlivé operace zásobníku neimplementujte, předpokládejte, že jsou hotovy, a pouze je vhodně volejte.

14.

- Merge Sort lze naprogramovat bez rekurze („zdola nahoru“) také tak, že nejprve sloučíme jednotlivé nejkratší možné úseky, pak sloučíme delší úseky atd. Proveďte to.