Egzamin AiSD 2013 - Odpowiedzi do pierwszej części

1. (KMR) Jak numerować słowa 17-literowe mając numerację 16-literowych?

> **Rozwiązanie:** Niech #16(n) - numery przypisane słowom 16-literowym.
> Mamy znaleźć numerację słów 17-literowych.
> W takim wypadku tworzymy sobie pary (#16(k), #16(k+1)) dla każdego k od 1 do n-1.
> Po tym kroku sortujemy te pary (leksykograficznie). Najlepiej wykorzystać do tego sortowanie kubełkowe.
> Po takiej operacji ustalamy kolejne numery od 1 dla par. Nowe numery są numeracją słów 17-literowych.

2. Jak uogólnić wykładowy algorytm szukania izomorfizmów drzew ukorzenionych na drzewa nieukorzenione. Jaką ma złożoność nowy algorytm?

> **Rozwiązanie:** Ukorzeniamy drzewa w centroidzie. Znalezienie centroidu robi się liniowo, np. BFS-em. Dalej stosujemy algorytm wykładowy.

3. Maksymalna liczba rotacji przy usuwaniu klucza z drzewa AVL

> **Rozwiązanie:** W przypadku usuwania klucza z drzewa AVL może się wydarzyć sytuacja, w której wskaźniki balansu będą zepsute aż do korzenia. Zatem maksymalna liczba rotacji to 2*log(n).

4. Po co rozróżnienie na pola nieużywane/usunięte przy haszowaniu z mapowaniem otwartym?

> **Rozwiązanie**: Haszowanie z mapowaniem otwartym porusza się po ciągu. Załóżmy, że szukamy D. Idąc funkcją haszującą przechodzimy przez ciąg A -> B -> C -> D. OK. Wywalmy B ze słownika. Mamy teraz A -> (puste) -> C -> D. Ponieważ każdy taki ciąg kończy się znakiem pustym (i nie rozróżniamy pustych pustych i pustych usuniętych), uznamy, że ten składa się tylko z A i nie odnajdziemy nigdy C i D.

5. Złożoność funkcji pi (z KMP)

> **Rozwiązanie:** Złożoność funkcji pi to O(n).
> Wynika to z następujących obserwacji:
> Wartość pi(k) < k.
> Wartość pi(k) zwiększa się maksymalnie o 1 w każdym kroku algorytmu.
> Wartość pi(k) może spaść co najmniej o 1 i tyle razy, ile zwiększyliśmy razy wartość.

6. Jak powinna wyglądać struktura wierzchołka w drzewie dwumianowym (jakie pola ma

zawierać)

Rozwiązanie: Minimalistycznie to będzie pamiętanie syna i prawego brata (na lewo syn, na prawo brat) plus jakaś komórka na pamiętany klucz.

7. Przykłady dwóch problemów NP-zupełnych (innych niż cykl hamiltona)

Problem kliki / zbioru niezaleznego w grafie (antyklika)
Problem spełnialnosci formuły logicznej
Problem pokrycia wierzchołkowego grafu nieskierowanego
Problem sumy podzbioru(czy podzbiór sumuje sie do zera)
Problem komiwojażera
Problem kolorowania grafu

8. Pseudokod dla operacji `insert` w drzewcu

```
INSERT-TREAP(k, p, T):
    Utwórz nowy węzeł x
    key[x] ← k
    priority[x] ← p
    INSERT-BST(x, T)
    while x ≠ root[T]
        if left[parent[x]] = x
            if priority[parent[x]] > priority[x]
                break
            else
                rotate-right(x)
        else
            if priority[parent[x]] > priority[x]
                break
            else
                rotate-left(x)
```

http://informatyka.wroc.pl/sites/default/files/user_files/u387/treap8_0.jpg

9. Wskazać i uzasadnić błąd w opisie FFT

a) oblicz wartosci wielomianow A i B w w1...w2n - pierwiastkach z jednosci stopnia 2n, oznaczmy te wartosci przez yi oraz zi (dla i= 0...2n - 1) ,
b) wymnoz te wartosci tj. ti = yi * zi; (dla i= 0...2n - 1)
c) uzywajac wzorów interpolacyjnych Lagrange oblicz wspolczynniki wielomianu stopnia 2n - 2 o wartosciach ti w w0, ... , w2n -1

Rozwiązanie: Interpolacja Lagrange'a działa w czasie kwadratowym (więc pierwsze F w

FFT nie zachodzi), do tego Lagrange'a się nie nadaje do zespolonych.

A jak powinien wyglądać prawidłowy opis? Tam były jakies specjalne wielomiany...

10. Ile może być maksymalnie drzew w kopcu dwumianowym eager/lazy

Rozwiązanie: W kopcu dwumianowym lazy maksymalna liczba drzew to n drzew (wyjdziemy z sytuacją, gdzie mamy n drzew 1-elementowych).

11. W algorytmie wyznaczania mediany przez próbkowanie ze zbioru S wybiera się najpierw zbiór R. Jaki jest jego maksymalny rozmiar (procentowo względem S)

12. Jaka jest oczekiwana maksymalna długość listy w haszowaniu liniowym w słowniku o N miejscach jeśli haszujemy N kluczy?

Rozwiązanie: Po pierwsze, haszowanie liniowe = haszowanie z adresowaniem otwartym gdzie f. haszującą mamy stworzoną za pomocą metody liniowej (czyli z h(x) robi sie h(x, i) = (h(x) + i) mod m.

13. Maksymalna liczba odwołań do pamięci zewnętrznej przy B-drzewie 1000/2000

Odpowiedź: log_1000(n).

14. Jak i po co stosuje się zaokrąglanie rozwiązania programu liniowego?

15. Dolna granica liczby porównań dla scalenia dwóch ciągów n-elementowych (n+m-1)

16. Do czego służy algorytm Strassena i jaka jest jego złożoność?

Mnożenie macierzy, $O(n^3)$ (ale nie $\Theta(n^3)$!!!), $\sim\Theta(n^{2.81})$
http://en.wikipedia.org/wiki/Strassen_algorithm

17. Jak szybko można znaleźć najdłuższy rosnący podciąg

Odpowiedź: Mozemy wykorzystac nastepujacy algorytm dynamiczny:

1. A ← "wektor wektorów"
2. Dla każdego elementu z ciągu:
    a. Znajdź jego miejsce w (wektor) w najdłuzszym podciągu jego zawierającym (mozemy zrobić to używając wyszukwiania binarnego na tablicy), taki że ostatni jego element jest pierwszym od niego większym w ciągu i dodaj go na koniec wektora. Jeżeli takiego wektora nie ma (element dodawany jest największy), to dodaj nowy wektor na końcu A.
3. LIS to ciąg pierwszych elementów z każdego wektora w wektorze A

Czas: $\Theta(n \lg n)$
http://informatyka.wroc.pl/node/402?page=0,1 - tutaj jaśniej
Rozwiązanie inne (tak samo szybkie):
Najpierw przeskalujmy dane tak, aby mieć je z zakresu od 1 do N (tysiące sposobów na zrobienie tego szybko, ot choćby skopiowanie ciągu gdzieś indziej, posortowanie go, potem przejście oryginalnego ciągu i wyszukanie binarne w tej posortowanej tablicy) - da się to zrobić jeszcze na inne sposoby.
Teraz programowanie dynamiczne (załóżmy, że chcemy tylko długość, odzyskanie

samego ciągu jest takie jak zwykle, dla każdego stanu pamiętamy skąd wzięliśmy dla niego wynik i cofamy się po tym): stanem jest dp[x][y] - długość LISa dla prefiksu x elementowego, przy założeniu, że kończymy na elemencie o wartości y. I teraz znając dp[x][] aby policzyć dp[x+1][] możemy, albo wziąć element T[x+1], albo nie -> nie wzięcie = przepisanie wyniku z dp[x], wzięcie = wzięcie max(dp[x][p]) + 1, gdzie p jest mniejsze od T[x] (ostatnio użytym elementem przed tym, który właśnie chcemy użyć musi być element mniejszy od niego) -> jak szybko znaleźć takie maksimum? Drzewo przedziałowe (dokładnie takie samo jak przy okazji zadania z chmurami na pracowni)
To rozwiązanie może nie jest przyjemne w implementacji, ale jest proste w zrozumieniu (mam nadzieję)
Warto wspomnieć, gdyby kogoś interesowało, że rozwiązanie pierwsze (przed rozwiązaniem innym) jest także programowaniem dynamicznym - tylko z odwróconym pomysłem: dla ustalonego wyniku (długości podciągu) pamiętamy najmniejszy element, którym może się kończyć podciąg rosnący o zadanej długości. Ta uwaga może się przydać w lepszym zrozumieniu lub udowodnieniu poprawności.

18. Jakiego rodzaju kopca użyć dla asymptotycznie najefektywniejszego zaimplementowania algorytmu a) Prima, b) Dijkstry

b)Prima - kopiec fibbonaciego, zlozonosc: O(E + V*logV)
b)dijkstry - kopiec fibbonaciego, zlozonosc: O(E + V*logV)

19. Jak działą sieć półczyszcząca (i jej wykorzystanie) w sortowaniu ciągów bitonicznych

20. Jak równolegle mnożyć liczby binarne w czasie O(log n)