Algorytmy i Struktury Danych (2023)

Lista zadań 9 (programowanie dynamiczne)

1. Znajdź (bez użycia komputera) optymalne nawiasowanie iloczynu macierzy, których wymiary tworzą ciąg [5, 10, 2, 12, 5, 50, 6]. Spamiętywanie (jeśli go użyjesz) wykonuj na kartce.

2. Podane w załączniku (oraz na `panoramixie`) programy `progdyn1.cc` oraz `progdyn2.cc`, rozwiązujące problem optymalnej triangulacji oraz optymalnego nawiasowania z wykorzystaniem spamiętywania, przepisz w taki sposób, by procedury znajdowania minimum nie korzystały z rekurencji, tylko wypełniały tablicę wyników w pętli w takiej kolejności, by w momencie gdy wyliczana jest wartość `F[i][j]` wszystkie potrzebne do jej wyliczenia elementy tablicy były już wypełnione. Wskazówka: należy wyliczać najpierw te pary `F[i][j]` dla których różnica `j-i` jest najmniejsza, czyli najpierw wyliczyć wszystkie koszty mnożenia dwóch sąsiednich macierzy, potem trzech, itd.

3. Jaka jest złożoność procedury wyznaczającej koszt optymalnego mnożenia ciągu n macierzy, zaimplementowana metodą z poprzedniego zadania, czyli wstępującą (bez rekurencji). Ile mnożeń wykona ta procedura? (a) w notacji O. (b) dokładnie.

4. (0.5pkt) Udowodnij, że wszystkich sposobów pocięcia pręta długości n, na kawałki całkowitej długości, jest 2^{n-1}.

5. (2pkt) Dany jest długi pręt stalowy długości n będącej całkowitą liczbą centymetrów, oraz tablica `double cena[n+1];` taka, że `cena[i]` określa, za ile można sprzedać kawałek długości i cm dla każdego $1 \leq i \leq n$ (kawałki długości zero są darmowe :). Napisz regułę rekurencyjną pozwalającą obliczyć jak pociąć na kawałki całkowitej długości pręt, by najwięcej na tym zarobić:
   (a) zakładając że cięcie jest darmowe
   (b) zakładając, że każde przecięcie pręta kosztuje c zł.

6. Zastosuj programowanie dynamiczne, by napisać program, który rozwiązuje problem z poprzedniego zadania: drukuje rosnącą listę długości kawałków, oraz całkowity zysk (dochód ze sprzedaży, pomniejszony w punkcie (b) o koszt cięcia).

   Wskazówka: procedura wypełnia tablice `MaxGain` oraz `LongestBit` (albo `ShortestBit`).

7. (2pkt) Optymalne drzewa poszukiwań binarnych. Istnieje wiele drzew BST zawierających dokładnie ten sam (uporządkowany rosnąco) zestaw kluczy. Zastosuj programowanie dynamiczne do znalezienie optymalnego drzewa poszukiwań binarnych (patrz Cormen). Załóż, że koszt wyszukania klucza to liczba węzłów, jakie trzeba odwiedzić, by go znaleźć (dla klucza w korzeniu koszt 1, piętro niżej – koszt 2, itd). Dane do zadania stanowi tablica `ile`, taka, że `ile[i]` mówi ile razy będzie wyszukiwany i-ty co do wielkości klucz.