Homework #3

Assigned: Tuesday, March 23, 2004
Due:
  - On-campus: Thursday, April 1, 2004 by 4 PM EST
  - FEEDS / NTU: Thursday, April 8, 2004 by 5 PM EDT
Total points: 100

Question 1: (20 points)
You are a librarian and you need to arrange \( n \) books (already sorted by ISBN numbers, assume 1 through \( n \) for your convenience) on a tall bookshelf (consisting of horizontal racks of pre-defined fixed width \( L \)) so that books appear in ascending order of their ISBN number from left to right, top to bottom. You are given the width data for all these books, and every width value is an integer (in millimeter), and you must keep a 1-mm thick separator between every two books. No separator is required for a book which is on the leftmost position of a rack. However, any vacant space on the right must be filled up with an expensive packing material, whose cost is proportional to the cube of the thickness of the packing pad. No packing material is needed for the bottommost rack (where you finish putting the books). Assume that there are enough number of racks in the shelf.

As an example, suppose you choose to put 6 books of width 1, 4, 3, 8, 2 and 4 (rack width = 25), then you need to put four 1-mm thick separators between the first five books, and need to fill up the rest of space with a 3-mm padding. In the rack below it, you place the last book and your total cost is 27 units.

Give an efficient algorithm (in terms of \( n \) and \( L \)) to arrange the books so that the cost is minimum.

Question 2: (10 points)

Give an algorithm to visit ALL the nodes in a graph by visiting every edge exactly twice. Using the result, explain how you can enter a maze (a menacing building consisting of thousands of paths where people get lost) and come out alive if you only have a marker using which you can mark the walls of maze (every path in the maze has two walls defining it).

Question 3: (20 points)

You are given a linear row of cattle ranches of sizes \( S[1 \ldots n] \). The objective is to specify merging operations so that all of the ranches are merged into a single giant cattle ranch. The merging rules are as follows:

- At any step, only a pair of consecutive ranches can be merged into a new ranch that is the physical union of the two ranches.
- The size of the new ranch is the sum of the sizes of the two ranches being merged.
• The cost of a merge is equal to the size of the “larger” of the two ranches being merged. The total cost of a sequence of \(n-1\) merges is the sum of the cost of the individual merges.

The goal is to give an efficient algorithm that computes the minimum total cost for creating one giant ranch from the original \(n\) ranches.

a) Let \(C(i,j)\) be the “minimum cost” of merging ranches \(i\) through \(j\). Write a recurrence relation for \(C(i,j)\).

b) Use the recurrence relation above to devise an efficient (polynomial time) algorithm to solve for the minimum cost for merging all the ranches.

c) Analyze your algorithm and determine its time complexity.

**Question 4: (10 points)**

A bipartite graph is a graph \(G = (V,E)\) is an undirected graph whose vertices can be partitioned into two disjoint sets \(V_1\) and \(V_2 = V - V_1\) with the properties that no two vertices in \(V_1\) are adjacent in \(G\) and no two vertices in \(V_2\) are adjacent in \(G\).

Write an algorithm to determine whether a graph \(G\) is bipartite. If \(G\) is bipartite, your algorithm should obtain a partitioning of the vertices into two disjoint sets \(V_1\) and \(V_2\) satisfying the properties above. Show that if \(G\) is represented by its adjacency lists, then the algorithm can be made to work in time \(O(n + e)\), where \(n=|V|\) and \(e=|E|\).

The graph in the figure is bipartite. A possible partitioning of \(V\) is \(V_1 = \{1, 4, 5, 6, 7\}\) and \(V_2 = \{2, 3, 8\}\).

**Question 5: (20 points)**

Following is a description of the longest monotonically increasing subsequence problem.

Given a sequence of numbers \(X = <x_1, x_2, ..., x_n>\) a monotonically increasing subsequence is any sequence \(X_0 = <x_{a_1}, x_{a_2}, ..., x_{a_k}>\) such that \(1 \leq a_1 < a_2 < ... < a_k \leq n\) and \(x_{a_1} \leq x_{a_2} \leq ... \leq x_{a_k}\). The goal is to find the length of the longest monotonically increasing subsequence. For example let \(X = <1, 4, 3, 5, 17, 10, 15, 12, 20>\), then \(X_0 = <1, 3, 5, 10, 15, 20>\) is one of the possible longest monotonically increasing subsequences. The length of this sequence is 6. Note that all the \(a_i\)’s denote the indices in the sequence \(X\).

(a) We will first solve a restricted sub-case of the above problem: Given a sequence \(X = <x_1, x_2, ..., x_n>\) find \(L(n)\), the length of the longest monotonically increasing subsequence that ends with \(x_n\).

Prove that for each \(k \in \{1, ..., n\}\), the following relation holds:

\[
L(k) = 1 + \max_{j \in \{1, ..., k-1\} \text{ with } x_j \leq x_k} L(j)
\]
(b) Use the above formulation to write an algorithm using dynamic programming that solves for \( L(n) \) in \( O(n^2) \) time.

(c) Show how you can use this algorithm to find the length of the longest monotonically increasing subsequence without the restriction that it ends with \( x_n \).

**Question 6: (20 points)**

We want to write a sentence on the floor using **prefabricated** tiles. Unfortunately, we cannot buy tiles with single letters and we cannot write all sentences with the available tiles: See Figure below for an example.

![Example Figure]

Given a sentence \( S \) of length \( n \) and a set of \( m \) tile types \( T = \{ t_0, t_1, \ldots, t_{m-1} \} \) we want to decide if it is possible to write \( S \) (assuming an unlimited number of each tile type). We can solve the problem with the following procedure (using the call \( \text{Write}(0,n-1) \)):

\[
\begin{align*}
\text{Write}(i,j) & \quad \text{IF } i > j \text{ THEN return TRUE} \\
& \quad \text{FOR } k = 0 \text{ to } m - 1 \text{ DO} \\
& \quad \quad \text{IF } S(i \ldots j) = t_k \text{ THEN return TRUE} \\
& \quad \text{FOR } l = i \text{ to } j - 1 \text{ DO} \\
& \quad \quad \text{IF Write}(i,l) \text{ AND Write}(l + 1, j) \text{ THEN return TRUE} \\
& \quad \text{return FALSE} \\
\end{align*}
\]

END Write

Here \( S(i \ldots j) \) denotes the sub-sentence from character \( i \) to character \( j \) (including both the characters). We assume that the test \( S(i \ldots j) = t_k \) take time \( O(j - i + 1) \).

a) Show that the running time of the above algorithm is \( \Omega(2^n) \).

b) Design a more efficient algorithm to solve the problem.

c) Analyze and give an appropriate asymptotic bound for the running time of your algorithm.