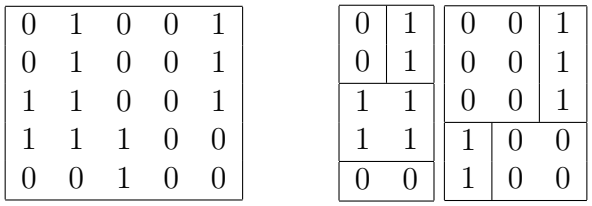
Algorithms (CS2443) : Problem Set 2

1. (0-1 knapsack) A thief enters a shop with a bag that can hold W kg. He sees a number of gold bars whose value his trained eye can estimate accurately. He must steal a subset of the bars so that they do not exceed the capacity of the bag and so that their value is maximum. Describe an efficient algorithm that solves this problem after accepting the number n of bars, their values and their weights as input. State the running time of your algorithm in terms of n and W.

2. We call a sequence $X_1, X_2, \ldots$ of numbers as "accelerating" if for every $i \geq 2$ we have $X_{i+1} - X_i > X_i - X_{i-1}$. For example, the sequences $1, 3, 6$ and the sequence $1, -4, 0, 5$ are accelerating, whereas the sequences $1, 3, 5$ and $-5, 4, 0, 3$ are not accelerating. A sequence of length less than 3 is considered to be accelerating. Describe an efficient algorithm to find the longest accelerating subsequence of an array $A[1, 2, \ldots, n]$.

3. You are given a rectangular sheet of paper filled with a black-and-white image. Your goal is to cut it into smaller sheets of paper each filled completely with black or completely with white. Every cut that you make must completely divide one sheet into two smaller rectangular sheets. The goal is to minimize the number of cuts. The input is represented as a $m \times n$ matrix of 0s and 1s, with 0 standing for Black and 1 for White. An example is shown below, with the input on the left and a legal sequence of 7 cuts on the right, with the first cut being made after the second column.



   Give an efficient algorithm for the problem stated above.

4. Suppose you are given a sequence of integers separated by + and - signs; for example:

$$1 + 3 - 2 - 5 + 1 - 6 + 7$$

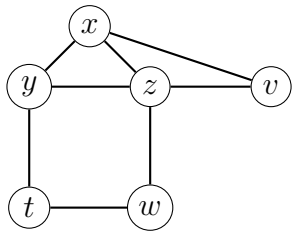
   You can change the value of this expression by adding parentheses in different places. For example:

$$((((((1 + 3) - 2) - 5) + 1) - 6) + 7) = -1$$
$$(1 + (3 - (2 - 5))) + ((1 - 6) + 7) = 9$$
$$((1 + (3 - 2)) - (5 + 1)) - (6 + 7) = -17$$

   Describe and analyze an algorithm to compute, given a list of integers separated by + and - signs, the maximum possible value the expression can take by adding parentheses.

Parentheses must be used only to group additions and subtractions; in particular, do not use them to create implicit multiplication as in $1 + 3(-2)(-5) + 1 - 6 + 7 = 33$.

5. A matching in a graph is a set of edges that do not share an end-point. For example, in the graph shown below, the set $\{xy, zw\}$ is a matching of size 2. A maximum-sized matching of this graph is $\{xv, yz, tw\}$.



   Describe an $O(n)$ algorithm to find a maximum-sized matching when given a tree of n vertices as the input.

6. There are n different items to be purchased, each costing Rs.1000. The prices of these items increase periodically, with item i increasing by a factor of $r_i > 1$ every month; that is, after k months, the price of item i will be $1000 \cdot r_i^k$ rupees. Unfortunately, only one item can be purchased per month, so that most items must be bought at a higher than initial price. The goal is to find an optimal sequence in which to buy the items so that the total cost is minimized. Describe and analyze an efficient algorithm for this problem.

7. Recall the activity selection problem in which the input is a collection of intervals (representing start and finish times of activities) and we needed to pick the maximum number of non-overlapping activities.

   In class, we saw a greedy algorithm where we picked the activity with earliest finish time. For each of the following alternative greedy strategies, either prove that the resulting algorithm always constructs an optimal schedule, or describe a small example for which the algorithm does not produce an optimal schedule. Assume that all algorithms break ties arbitrarily [Hint: Three of these algorithms are correct.]

   (a) Choose an interval x that *ends last*, discard intervals that conflict with x, and recurse.

   (b) Choose an interval x that *starts first*, discard all intervals that conflict with x, and recurse.

   (c) Choose an interval x that *starts last*, discard all intervals that conflict with x, and recurse.

   (d) Choose an interval x with *shortest duration*, discard all intervals that conflict with x, and recurse.

(e) Choose an interval x that *conflicts with the fewest other intervals*, discard all intervals that conflict with x, and recurse.

(f) If no intervals conflict, choose them all. Otherwise, discard an interval with the *longest duration* and recurse.

(g) If no intervals conflict, choose them all. Otherwise, discard an interval that conflicts *with the most other intervals* and recurse.

(h) Let I_1 be the interval with the *earliest start time*, and let I_2 be the interval with the *second earliest start time*.

- If I_1 and I_2 are disjoint, choose I_1 and recurse on everything but I_1.
- If I_1 completely contains I_2, discard I_1 and recurse.
- Otherwise, discard I_2 and recurse.

(i) If any interval I_1 completely contains another interval, discard I_1 and recurse. Otherwise, choose the interval I_2 that *ends last*, discard all intervals that conflict with I_2, and recurse.