Exercises - Divide and Conquer (2) and RP - Algoritmiek

Tutorial February 8, 2024

1. **From A to B:** Suppose we are given integers a, b such that $2 \leq a \leq b$. Consider two operations: *Increment* (I), which increments a number by 1, and *Double* (D), which doubles a number. You are asked to find the minimum number of operations to get from a to b. For example, you can get from 8 to 38 in 4 steps (IDID).

   (a) Determine the list of choices and top choice for this problem.

   (b) Formulate a clear subproblem.

   (c) Give a recurrence for your top choice and subproblem.

   (d) Prove the optimality principle for your top choice and subproblem.

   Solution:

   (a) The list of choices are the choices I or D. The top/last choice is I (from $b-1$) to D (from $b/2$).

   (b) The subproblem is, given an integer c with $a \leq c \leq b$, to find the smallest number of operations to get from a to c.

   (c) $K(c)$ is the smallest number of operations to get from a to c. $K(a) = 0$. $K(c) = \min\{1 + K(c/2), 1 + K(c-1)\}$ if $c \geq 2a$. $K(c) = 1 + K(c-1)$ otherwise.

   (d) Consider a sequence of operations $S = \{o_1, \ldots, o_k\}$ to get from a to c with k as small as possible. Say $o_k = D$. Then $S \setminus \{o_k\} = \{o_1, \ldots, o_{k-1}\}$ is a solution to get from a to $c/2$. Let $T = \{p_1, \ldots, p_{k'}\}$ be an optimal solution for the subproblem with $c' = c/2$, so a way to get from a to $c/2$ with k' as small as possible. Then $k' \leq k-1$, because $S \setminus \{o_k\}$ is a solution. Note that $S' = \{p_1, \ldots, p_{k'}, o_k\}$ is a valid way to get from a to c, because T gets you from a to $c/2$ and $o_k = D$ get you from $c/2$ to c. Since $k' \leq k-1$, S' contains at most k operations, and thus is again an optimal solution. Can you argue the same when $o_k = I$?

   This answer assumes you go 'big to small'. You can also go 'small to big'. (In other words, there are two ways of looking at the solution of this problem. Either will be correct.)

2. **Exact change: alternative method:** For the Exact Change problem, we can also consider the following list of choices: how many coins of value a_1 do we return, how many coins of value a_2, etc.

   (a) Define the top-choice and the subproblem you need to use. Does one parameter for your subproblem suffice?

   (b) Give the recurrence for this formulation of the problem. Don't forget the base case!

   (c) Prove that the optimality principle holds.

   (d) Discuss the relation of this recurrence with the recurrence discussed in the lecture.

   Solution:

   (a) The top-choice is the number of coins of value a_r you return. (The number of coins of value a_1 as a top-choice might also lead to a correct solution!) The subproblem is, given integers c ($0 \leq c \leq b$) and i ($1 \leq i \leq r$), what is the smallest number of coins needed to pay exactly c using coins of value $a_1, \ldots, a_i$?

(b) Recurrence: for integers c ($0 \leq c \leq b$) and i ($1 \leq i \leq r$), P[i,c] = smallest number of coins needed to pay exactly c using coins of value $a_1, \ldots, a_i$.
Don't forget to explicitly state what $P[i, c]$ means.
$P[i, 0] = 0$ for all $1 \leq i \leq r$
$P[1, c] = c/a_1$ for all $1 \leq c \leq b$ for which a_1 is a divisor of c
$P[1, c] = \infty$ for all other values of $c > 0$
$P[i, c] = \min_{0 \leq k \leq c/a_i}\{P[i-1, c-ka_i] + k\}$ for $1 \leq c \leq b$, $2 \leq i \leq r$

(c) Consider an optimal solution P to the subproblem for parameter i that pays $p_1, \ldots, p_i$ coins of value $a_1, \ldots, a_i$ to pay out c. Consider any integer i' with $1 \leq i' < i$ and let $d = c - \sum_{j=i'+1}^{i} p_j$. Note that P pays $p_1, \ldots, p_{i'}$ coins of value $a_1, \ldots, a_{i'}$ to pay out d. Let Q be an (optimal) solution to the subproblem for d and i'; that is, Q pays $q_1, \ldots, q_{i'}$ coins of value $a_1, \ldots, a_{i'}$ to pay out d. Note that $q_1, \ldots, q_{i'}, p_{i'+1}, \ldots, p_i$ yields a valid solution to pay out c. Moreover, since Q uses at most as many coins as $P[1, i'] = p_1, \ldots, p_{i'}$ by the definition of the subproblem ($\sum_{j=1}^{i'} q_j \leq \sum_{j=1}^{i'} p_j$), the solution $q_1, \ldots, q_{i'}, p_{i'+1}, \ldots, p_i$ uses at most as many coins as P ($\sum_{j=1}^{i'} q_j + \sum_{j=i'+1}^{i} p_j \leq \sum_{j=1}^{i} p_i$) and is optimal too! Note that it suffices to do this proof for $i' = i - 1$.

(d) It is the exact same idea, except we allow to pay out k coins of a particular value at once. In the recurrence in the lecture, uses of (say) coin 1 and 2 can alternate, but here this cannot happen.

3. **Windmills:** Windmills are being built all along the Dutch coast. There are n possible locations to build windmills, say on a line, where location i lies 1km to the north of location $i-1$. Due to wind conditions, by building a windmill at location i you harvest energy e_i. To not disrupt nature and windflow, windmills must be built at least K kilometers apart, for some integer $K \geq 1$. Subject to this condition, what is the maximum amount of wind energy you can harvest by appropriately placing windmills?

(a) Determine the sequences of choices made and define the top-choice and subproblem.

(b) Give a recurrence following your subproblem and top-choice. Don't forget the base case!

(c) Prove that the optimality principle holds.

Solution:

(a) The sequence of choices is whether or not to build a windmill at each location i, $i = 1, \ldots, n$ (the locations on which to build the windmills leads to a different but equally correct answer to this exercise; see below). The top-choice is whether or not to build a windmill on the last location. The subproblem is, given an integer i ($1 \leq i \leq n$), what is the maximum amount of wind energy you can harvest by placing a windmills on a subset of locations $1, \ldots, i$ such that windmills are at least K km apart. Note that the general problem is a special case of the subproblem (choose $i = n$).

(b) $P[i]$ is the maximum amount of wind energy you can harvest by placing windmills on a subset of locations i such that they are at least K km apart.
Be sure to define what you mean by $P[i]$!
$P[1] = e_1$; $P[i] = \max\{e_i + P[i-K], P[i-1]\}$ for $i = 2, \ldots, n$.

(c) Let $O \subseteq \{1, \ldots, i\}$ be an optimal solution of the placement of windmills for a subproblem with parameter i.
Suppose $i \in O$. Let P be an (optimal) solution to the subproblem for parameter $i - K$. Let $S = P \cup \{i\}$. Note that S is a valid solution, because all windmills of P have distance at least K to i. The windmills placed by P yield at least the same amount of energy as the windmills placed by O in the first $i - K$ locations ($\sum_{j \in P} e_j \geq \sum_{j \in O \cap \{1,\ldots,i-K\}} e_j$). As both S and O place a windmill at i, S yields at least as much energy as O, and thus S is also an optimal solution.
Suppose that $i \notin O$. Let P be an (optimal) solution to the subproblem for parameter $i-1$. Note that P is a valid solution. The windmills placed by P yield at least the same amount of energy as the windmills placed by O in the first $i - 1$ locations ($\sum_{j \in P} e_j \geq \sum_{j \in O \cap \{1,\ldots,i-1\}} e_j$). As neither P nor O place a windmill at i, P yields at least as much energy as O, and thus P is also an optimal solution.
Hence, the optimal solution is constructed from a combination of optimal solutions to the subproblem.

Alternate solution

(a) The sequence of choices is the locations on which to build the windmills. The top-choice is the last location where a windmill is built. The subproblem is, given an integer i $(1, \ldots, n)$, what is the maximum amount of wind energy you can harvest by placing a windmill on location i and windmills on locations $1, \ldots, i - K$ that are at least K km apart. Note that the general problem is a special case of the subproblem by considering the best solution to a subproblem for $i = 1, \ldots, n$.

(b) $P[i]$ is the maximum amount of wind energy you can harvest by placing a windmill on location i and windmills on locations $1, \ldots, i - K$ that are at least K km apart.
Be sure to define what you mean by $P[i]$!
$P[i] = e_i$ for $i = 1, \ldots, K$; $P[i] = e_i + \max_{1 \leq j \leq i-K} P[j]$ otherwise.

(c) Let $O \subseteq \{1, \ldots, i\}$ be an optimal solution of the placement of windmills for a subproblem with parameter i. Let $i' \in O$ be a location where a windmill is placed. Let P be an (optimal) solution to the subproblem for i'. Let S be constructed by combining P and what remains from O after removing windmills on locations $1, \ldots, i'$ ($S = P \cup (O \setminus \{1, \ldots, i'\})$). Note that S is still a valid solution: placing a windmill at i' does not conflict with the remainder of O (O itself placed a windmill at i') and all windmills of P are at least distance K apart and distance at least K from any windmill in the remainder of O. Moreover, the windmills placed by P yield at least the same amount of energy as the windmills placed by O in the first i locations ($\sum_{j \in P} e_j \geq \sum_{j \in O \cap \{1,\ldots,i'\}} e_j$). As both S and O place a windmill in location i, S yields at least as much energy as O, and thus S is also an optimal solution.

4. **Dominatie in lineaire tijd:** Consider the domination ("Beste Koop") problem from the lecture. Give an $O(n)$ time, divide & conquer algorithm for this problem. You may assume the input is already sorted on quality. Give the algorithm, the recurrence for its running time, and prove that the recurrence is bounded by $O(n)$.

   Solution: The crux to getting to $O(n)$ total running time is to get rid of the $O(n)$ term in every call to the algorithm given in class. The $O(n)$ is there due to the minimum finding. However, we know that minimum finding can be done in a divide & conquer way. So every recursive call returns not only the best product, but also the minimum on its side. The minimum of the left side can then be used as an input to the right side to perform the filtering.

   ```
   Method Dom2(product[] X, int a, int b, int p*) {
   // assumption: X sorted on quality
   // p* will be the price on which we filter
   // return (Pareto-optimal products, minimum price)
   if a+1==b and X[a].prijs < p*
   then return ({X[a]},X[a].prijs)
   // here we filter out prices that are too high
   if a+1==b and X[a].prijs ≥ p*
   then return ({}, X[a].prijs)
   // on the left side, we do not filter
   (R1,m1) = Dom2(X, a, (a+b)/2, ∞)
   // on the right side, we filter based on the minimum price of the left side
   (R2,m2) = Dom2(X, (a+b)/2, b, m1)
   return (R1 plus R2, min(m1,m2))
   }
   ```

   The running time is $T(n) = 2T(n/2) + O(1)$. Using the Master theorem, this solves to $T(n) = O(n)$.

5. **Mister Animal:** Mister Animal has a video channel and in his new video, he has the following challenge for an unsuspecting viewer, you! Mister Animal gives you B dollars to shop equipment in an electronics store. The store has n item of value $v_1, \ldots, v_n$; each item can be found only once in the shop. The challenge is to spend exactly B dollars; if you fail, you do not get to keep the electronics and instead get pelted with water balloons. Before you start your spending spree, you strategize and look for a solution that spends exactly b dollars.

   (a) Determine the list of choices and the top choice for this problem. Think again about the number of parameters you might need.

   (b) Formulate a clear subproblem.

   (c) Give a recurrence for your top choice and subproblem.

   (d) Prove the optimality principle for your top choice and subproblem.

   Solution:

   (a) The sequence of choices is whether or not to buy item i for each item i. Hence, the top-choice is whether or not to buy item i.

   (b) The subproblem is, given items $1, \ldots, i$ and a budget of D dollars, to decide whether one can spend exactly D dollars on items $1, \ldots, i$. Note that the general problem is a special case of this subproblem (choose $i = n$ and $D = B$).

(c) Define $S[i, D]$ as a boolean, which is true if and only if one can spend exactly D dollars on items $1, \ldots, i$.
S[0,0] = true
S[0,D] = false for all $0 < D \leq B$
S[i,D] = S[i-1,D] if $v_i > D$ and $i > 0$
S[i,D] = S[i-1,$D - v_i$] $\vee$ S[i-1,D]

(d) Suppose there is a solution $O \subseteq \{1, \ldots, i\}$ for a subproblem with parameters i and D; that is, $\sum_{j \in O} v_j = D$.
Suppose that $i \in O$. Then let $S \subseteq \{1, \ldots, i-1\}$ be a solution to the subproblem for $i-1$ and $D - v_i$ dollars. Such a solution must exist, because $O \setminus \{i\}$ is a potential solution. Then $S \cup \{i\}$ is a solution to spend exactly D dollars on a subset of items $\{1, \ldots, i\}$.
Suppose that $i \notin O$. Then let $S \subseteq \{1, \ldots, i-1\}$ be a solution to the subproblem for $i-1$ and D dollars. Such a solution must exist, because O is a potential solution. Then S is a solution to spend exactly D dollars on a subset of items $\{1, \ldots, i\}$.