Exercises - Greedy - Algoritmiek

Tutorial 22 February 2022

1. **Pancakes (part 1):** You have bought a Pancakebot, a 'printer' for pancakes. Your idea is to use it to put on children's parties. There are n potential children's parties you can bring your Pancakebot to, where party i has a fixed start time $s(i)$ and end time $e(i)$. Because you have only one Pancakebot, you have to select a subset of the parties to go to. Each party has the same profit, 100 euros. Which parties do you select to earn as much money as possible?
   We consider several strategies by which to select the parties. Each strategy proposes a rule to select a party; after a party is selected, we remove all parties that overlap in time with the selected party (i.e. that are not compatible) and re-apply the rule until no longer possible. This always yields a feasible solution.

   (a) Consider a strategy where you always select a party that starts earliest, that is, with the minimal start time $s(i)$. Give an example that shows that following this strategy might yield a sub-optimal solution.

   (b) Consider instead a strategy where you always select a party of shortest duration, that is, with minimal $e(i) - s(i)$. Give an example that shows that following this strategy might yield a sub-optimal solution.

   (c) Consider now a strategy where you always select a party that ends earliest, that is, with the earliest ending time $e(i)$. Use an exchange argument to argue that this strategy yields an optimal solution.
   Hint: consider an optimal solution that has the most parties in common with the greedy solution.

   (d) As an alternative argument, use induction to prove that greedy always "stays ahead". That is, $e(G_i) \leq e(O_i)$ for all $1 \leq i \leq k$, where $G_1, \ldots, G_k$ are the greedily selected parties and $O_1, \ldots, O_\ell$ is an optimal solution. Then show that $k = \ell$.

   Solution:

   (a) A single, very long party that starts earliest and overlaps many short parties.

   (b) Two long, non-overlapping parties, and a single short party that overlaps the end time of the first and the start time of the second.

   (c) Let $G_1, \ldots, G_\ell$ the parties selected by the greedy strategy and let $O_1, \ldots, O_k$ be the parties selected by an optimum solution, both numbered by increasing end time (so $e(O_i) < e(O_{i+1})$ for all $1 \leq i < k$ and $e(G_j) < e(G_{j+1})$ for all $1 \leq j < \ell$. Suppose the optimum solution has been selected such that it has the most parties that greedy also selected, so $|\{G_1, \ldots, G_\ell\} \cap \{O_1, \ldots, O_k\}|$ is maximum.
   Suppose they are different and let i be the first index where they differ. By the definition of i, $s(O_i) \geq e(O_{i-1})$ and $s(G_i) \geq e(G_{i-1}) = e(O_{i-1})$. By the choices of greedy, $e(G_i) \leq e(O_i)$. Hence, we can replace O_i by G_i and obtain a valid solution without overlaps. Also note that $e(G_i) \leq e(O_i) < e(O_{i+1}) < \cdots < e(O_k)$ and thus G_i is not one of $O_{i+1}, \ldots, O_k$. Hence, $O_1, \ldots, O_{i-1}, G_i, O_{i+1}, \ldots, O_k$ is also an optimum solution (the same number of parties). It has one more party in common with greedy, a contradiction.

   (d) Base: $i = 1$, follows by definition of the greedy strategy. IH: it holds for all $j < i$. Step: since $e(G_j) \leq e(O_j)$ for all $j < i$ and $e(O_{i-1}) < e(O_i)$, O_i is available to the greedy strategy after selecting $G_1, \ldots, G_{i-1}$. Then $e(G_i) \leq e(O_i)$ by definition. Note that $\ell \leq k$ by definition. Suppose $\ell < k$. As in the previous argument, O_k is available to the greedy strategy after selecting $G_1, \ldots, G_\ell$, a contradiction that G was constructed greedily.

2. **Pancakes (part 2):** Your idea to make pancakes using the Pancakebot is a great success! You decide to change your earnings model. For exactly 1 hour at party i, you ask an amount of k_i euros. Party i still has starting time s_i, which is guaranteed to be an integer. The duration is 1 hour, so the ending time of party i is $s_i + 1$. Which parties do you go to to make as much money as possible.

(a) Give an optimal greedy strategy and prove the greedy choice property using an exchange argument (uitwisselargument).

(b) Suppose there are n parties and $1 \leq s_i \leq n$. Give an algorithm to compute an optimal solution with running time $O(n)$.

Solution:

(a) For every start time, pick the party with maximum k_i. In an optimal solution, we can always replace the chosen party with the party chosen by the greedy solution.

(b) Use bucket sort and pick the maximum k_i in every bucket.

3. **Lampposts:** A long road has n houses, where house i has distance s_i from the start (you may assume the houses are sorted by distance, $i < j \Rightarrow s_i \leq s_j$). The city has decided to place lampposts along the road such that each house has a lamppost within distance M.
Example: If s is: 340, 670, 1200, 1600, 2400, 2710 and $M = 500$, then three lampposts suffice, for example on positions 500, 1500, 2400.
Give a greedy algorithm that computes in linear time the minimum number of lampposts needed. Prove the greedy choice property.

Solution: The last house needs to be lit. You can place the lamppost at distance M, so at $s_n - M$. Then we can prove that we always stay ahead. So if $g_m, \ldots, g_1$ is a placement of the lampposts by greedy and $o_k, \ldots, o_1$ is the placement of the optimum (here g_1 and o_1 are the last lampposts placed), then we can prove that $g_i \leq o_i$ for all i. Use induction. The base case is clear by the choice of g_1. Also, all houses after $g_1 - M$ and $o_1 - M$ are covered by greedy resp. optimum. IH: $g_{i-1} \leq o_{i-1}$ and all houses after $g_{i-1} - M$ and $o_{i-1} - M$ are covered by greedy resp. optimum. Note that $g_{i-1} - M \leq o_{i-1} - M$, so the first house left uncovered by greedy is at least as far to the left as the first house left uncovered by optimum. Then greedy can place the next lamppost at least as far to the left as the optimum, so $g_i \leq o_i$. Also, all houses after $g_i - M$ and $o_i - M$ are covered by greedy resp. optimum.
The algorithm now considers s_n, places a lamp at $s_n - M$, and then moves back to the largest i such that $s_i < s_n - M$, etc. Since the positions are sorted, this takes $O(n)$ time.

4. **Bicycle rental:** A rental shop has n bicycles. The height of bicycle j is f_j. De rental shop can rent all bikes to a group of n people. The height of person i is p_i. Because everyone likes to have a fitting bike, we calculate a *mismatch* of $|f_j - p_i|$ if person i rides bike j. Give a greedy algorithm that gives a bike to each person, while minimizing the sum of mismatches. Prove the greedy choice property! The running time of your algorithm should be $O(n \log n)$.

Solution: The greedy strategy is to give the tallest person the tallest bike. Suppose the bikes are sorted $f_1 \leq \cdots \leq f_n$. Consider greedy versus an optimum solution, and let b be the last bike for which they have different assignments. Say the optimum puts person o on b and greedy puts person g ($1 \leq o, g \leq n$). Note that $p_g \geq p_o$. Let a be the bike assigned to g in the optimum. Note that $f_a \leq f_b$. Then the current mismatch in the optimum for these two people is $|p_o - f_b| + |p_g - f_a|$. However, by the aforementioned inequalities,

$$|p_o - f_b| + |p_g - f_a| \geq |p_o - f_a| + |p_g - f_b|.$$

Hence, we can swap without increasing the sum of mismatches. Therefore, there exists an optimal solution that follows the greedy strategy.
For the algorithm, sort the bicycles and people by height. Then the assignment is straightforward. $O(n \log n)$ time.

5. **Fibonacci-Huffman:** Give an optimal Huffman-code for $\{a, b, c, d, e, f, g, h\}$, where the relative frequencies are as the Fibonacci-numbers:

σ	a	b	c	d	e	f	g	h
$f(\sigma)$	1	1	2	3	5	8	13	21

Solution: In this case, when applying Huffman's algorithm, any newly formed node after k steps always receives a frequency equal to the sum of the $k+1$ lowest Fibonacci numbers (so the sum of the first $k+1$ letters). This is because $\sum_{i=1}^{k+1} F(i) = F(k+3) - 1 < F(k+3)$. Then h=0, g=10, f=110, e=1110, d=11110, c=111110, b=1111110, a=1111111.