CS3230: Assignment for Week 9 Solutions

Due: Sunday, 3rd Apr 2022, 11:59 pm SGT.

---

1. (a) We claim that you should order the numbers in B so that $b_1 < b_2 < \cdots < b_n$. Indeed, suppose you order it in some other way; then there would be a pair (j,k) with $1 \le j < k \le n$ such that $b_j > b_k$. This means that $a_j^{b_j - b_k} < a_k^{b_j - b_k}$, and so $a_j^{b_j} a_k^{b_k} < a_j^{b_k} a_k^{b_j}$. Since all numbers are positive, your score $\prod_{i=1}^{n} a_i^{b_i}$ can be improved by switching b_j and b_k:

$$\prod_{i=1}^{n} a_i^{b_i} = \left(\prod_{i \notin \{j,k\}} a_i^{b_i} \right) \cdot a_j^{b_j} a_k^{b_k} < \left(\prod_{i \notin \{j,k\}} a_i^{b_i} \right) \cdot a_j^{b_k} a_k^{b_j},$$

so your ordering was not optimal. Hence, you can obtain an $O(n \lg n)$ algorithm by simply sorting all numbers in B, e.g., using Merge Sort.

(b) There cannot be a comparison-based algorithm running in time $o(n \lg n)$. Indeed, if there were such an algorithm, from the argument in part (a), we know that this algorithm must reorder the numbers in B in increasing order, so you would be able to sort the numbers in B using $o(n \lg n)$ comparisons, which is impossible.

2. (a) Let u be the unique neighbor of v. Suppose S is a maximum independent set of G. The set S must contain either u or v, because otherwise, $S \cup \{v\}$ is an independent set larger than S. Now, suppose S contains u but not v. Then, note that $(S \setminus \{u\}) \cup \{v\}$ is also an independent set because v is not a neighbor of any other node besides u. Hence, there is always a maximum independent set containing v.

(b) Let $N(v)$ denote the neighbors of a vertex v.

Claim 1 (Optimal Substructure)**.** *If v is contained in a maximum independent set S of G, then $S = \{v\} \cup S'$ where S' is a maximum independent set of the graph G' obtained by removing $v \cup N(v)$ from G.*

Proof. If v is contained in a maximum independent set S, then $N(v)$ cannot belong to S. The rest is a standard cut-and-paste argument: If there is a larger independent set S'' of G' than S', then $S'' \cup \{v\}$ is a larger independent set of G than S. $\square$

Combining the optimal substructure property with the greedy property from (a), we get the following algorithm: find a leaf v and include v in S, remove $\{v\} \cup N(v)$ from G, and repeat until graph is empty. The algorithm runs in $O(|V|)$ time, where V denotes the set of vertices in G.

3. (a) The following two claims establish the greedy property.

Claim 2. *In any optimal solution, for each $i < k$, there must be fewer than c coins of denomination d_i.*

Proof. For $i < k$, c coins of denomination d_i can be replaced by 1 coin of denomination d_{i+1}, reducing the number of coins by $c - 1$, so the solution would not be optimal. □

Claim 3. *Suppose $i^* = \max\{i : 1 \le i \le k, d_i \le n\}$. Then, any optimal solution must contain a coin of denomination d_{i^*}.*

Proof. The solution cannot contain any coin of denomination d_i for $i > i^*$ because $d_i > n$. For the sake of contradiction, suppose an optimal solution contains coins of denomination d_i for $i < i^*$ but not d_{i^*}. By Claim 2, there can be at most $c - 1$ coins of each of the denominations $d_1, \ldots, d_{i^*-1}$, so their total value is at most

$$(c-1) \cdot (c^0 + c^1 + \cdots + c^{i^*-2}) = c^{i^*-1} - 1 = d_{i^*} - 1,$$

which is a contradiction as $d_{i^*} - 1 < n$. □

Thus, by the greedy property above and the optimal substructure discussed in Lecture 8, we get the following algorithm: choose i^* as above, decrease n by d_{i^*}, and repeat until $n = 0$.

(b) Consider for the example the denominations $\{1, 5, 8\}$. If $n = 10$, the optimal solution is of size 2 (two 5-cent coins) while the greedy solution is of size 3 (one 8-cent and two 1-cent coins).