CSCI-GA.1170-001/002 Fundamental Algorithms November 24, 2015

Solutions to Problem 2 of Homework 9 (12 points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, November 24*

You have n CDs in your library labeled $1, \ldots, n$. You would like to arrange them on your CD rack in linear order $\pi(1), \ldots, \pi(n)$, according to some permutation π on n numbers. After such an arrangement is finalized, the cost to access CD i is equal to $\pi(i)$, as you need to scan through the first $\pi(i)$ CDs until CD i is found. Given a sequence of k CD requests $r_1, \ldots, r_k \in \{1 \ldots n\}$, your job is to figure out the permutation π (i.e., an array A such that $A[i] = \pi(i)$) having the *smallest total access cost* $\sum_{j=1}^{k} \pi(r_j)$.

(a) (4 pts) Describe in English a greedy algorithm for this problem.

Solution:

Greedy strategy is to pick that CD r_j that is most frequently used and place it on the top of the stack i.e assign corresponding $\pi(r_j)$ to be the least □

(b) (4 pts) Prove the correctness of your algorithm using the local-swap argument.

Solution:

Let the greedy algorithm algorithm G outputs the permutation π^G and the optimal algorithm Z outputs the permutation π^Z. Let r_m be the CD with maximum number of requests then the greedy algorithm sets $\pi^G_{r_m}$ to 1. Now there is some CD in π^Z whose value is set to 1 by Z i.e $\pi^Z_{r_p} = \pi^G_{r_m} = 1$. Now substitute $\pi^Z_{r_m}$ with $\pi^G_{r_m}$ then the resulting permutation will have two 1's in it. So substitute $\pi^Z_{r_p}(= 1)$ with the old value of $\pi^Z_{r_m}$. Let the resulting permutation be some π^{Z^*}. Then π^{Z^*} still remains to be optimal as clearly $\sum_{j=1}^{k} \pi^{Z^*}(r_j) \leq \sum_{j=1}^{k} \pi^Z(r_j)$. Therefore, the **Lemma** is, In the optimal solution the value of π for the CD with maximum number of requests is set to 1

Theorem If $\pi^Z = \pi_{r_m}, \pi^{Z`}$, where $\pi_{r_m} = 1$ i.e $r_m$ is the CD with maximum number of requests and $\pi^{Z`}$ is the optimal permutations of all the CDs starting from 2 i.e excluding r_m then we need to prove that π_z is also optimal.

Proof From the above Lemma we know that the optimal solution sets π_{r_m} to 1 i.e the CD with maximum number of requests is at the top of the stack so it means that π^Z might be an optimal solution. To show that it is indeed an optimal solution we need to prove that $\sum_{j=1}^{k} \pi^Z(r_j) \leq \sum_{j=1}^{k} \pi^{Z^*}(r_j)$.

Let $\pi^{Z^*} = \pi_{r_m}, \pi^{Z^{**}}$ then as $Z`$ is optimal $\sum \pi^{Z`} \leq \sum \pi^{Z^{**}} = \sum_{j=1}^{k} \pi^{Z*} - \pi_{r_m}$

$\implies \sum_{j=1}^{k} \pi^Z(r_j) = \pi_{r_m} + \sum \pi^{Z`} \leq \pi_{r_m} + \sum_{j=1}^{k} \pi^{Z*} - \pi r_m \leq \sum_{j=1}^{k} \pi^{Z*}$

Therefore, π^Z is an optimal solution. Hence the proposed greedy algorithm is correct □

(c) (4 pts) Write the pseudocode for implementation which runs in time $O(n + k)$. (Hint: At some point use counting sort.)

Solution:

```
1  Algorithm: ArrangeCDs(n, r_1, ..., r_k)
2  freq ← NewArray(n)
3  freq ← 0
4  for i ← 1 to k do
5  |  freq[r_i] ← freq[r_i] + 1
6  end
7  CD ← NewArrayTuples(n)
8  for i ← 1 to n do
9  |  CD(i) ← (i, freq[i]))
10 end
11 DescendingCountingSort_Tuples(CD)
12 counter ← 1
13 for i ← 1 to n do
14 |  π[CD[i](1)] ← counter
15 |  counter ← counter + 1
16 end
17 Return π
```

Algorithm 1: Greedy algorithm to minimize the total access cost of CDs in $O(n+k)$ time

In the above pseudocode $freq$ is an array such that $freq(i)$ maintains the frequency of the CD i's requests. This step takes $O(n)$ time.

CD is an array of tuples such that $CD(i)$ stores the tuple $(i, freq(i))$. Therefore creating this list of tuples CD takes $O(n)$ time.

Now that we have a list of CDs we need to sort them using their frequencies. DescendingCountingSort_Tuples is a variant of Counting Sort that sorts a list of tuples in descending order based on the second element of the tuple i.e it sorts all the CDs based on their request frequencies. The frequency of the CD requests range between 0 to k and there are total n number of CDs. Therefore this step takes $O(n+k)$ time

Now we have a list of CDs sorted in a decreasing order based on their frequencies. The last for loop iterates through these sorted CDs and assigns them the value of π in an increasing order. This step takes $O(n)$ time. Therefore the total running time of the algorithm is $O(n+k)$

□