Aditya Shirode, Rhythm Shah, Shrenuj Gandhi

2. [7 points, not evenly distributed]

   [from Brassard and Bratley, Fundamentals of Algorithmics, Prentice Hall, 1996]

   Suppose n programs are stored on a magnetic tape (remember those?). Let $s(i)$ be the size of program i and $f(i)$ the frequency of use for program i. The time it takes to access a program is proportional to the sizes of all programs on the tape up to and including it. The goal is to minimize the total access time for all programs by finding an ordering of the programs on the tape. More formally, let π be a permutation of $1, \ldots, n$ that describes the ordering: so if $\pi(1) = i$ then program i is the first on the tape. The total cost is:

   $$\sum_{i=1}^{n} f(\pi(i)) \sum_{k=1}^{i} s(\pi(k))$$

   There are three possible greedy algorithms for minimizing this cost:

   (a) select programs in order of increasing $s(i)$ (nondecreasing if we allow for equal sizes – I use increasing to avoid confusion)

   (b) select programs in order of decreasing $f(i)$

   (c) select programs in order of decreasing $f(i)/s(i)$

   For each algorithm, either find a counterexample to show that it is not optimal or prove that it always gives an optimum solution.

Answer:

(a) Based on sizes

i	s(i)	f(i)	f(i)/s(i)
1	2	4	2
2	4	8	2
3	6	16	2.66
4	8	32	4

In descending order based on $f(i)/s(i)$

i	s(i)	f(i)	f(i)/s(i)
1	8	32	4
2	6	16	2.66
3	4	8	2
4	2	4	2

Costs for above arrangements are as follows:

$$\begin{aligned} \text{cost(sizes)} &= 4 * (2) + 8 * (2 + 4) + 16 * (2 + 4 + 6) + 32 * (2 + 4 + 6 + 8) \\ &= 8 + 48 + 192 + 640 \\ &= 888 \end{aligned} \tag{1}$$

$$\begin{aligned} \text{cost}(f(i)/s(i)) &= 32 * (8) + 16 * (8 + 6) + 8 * (8 + 6 + 4) + 4 * (8 + 6 + 4 + 2) \\ &= 256 + 224 + 144 + 80 \\ &= 704 \end{aligned} \tag{2}$$

After comparing (1) and (2), we can see that descending ordering based on ratio of frequency to size gives a better performance than increasing ordering based on sizes.

Hence this greedy approach is not optimal.

(b) Based on frequency

i	s(i)	f(i)	f(i)/s(i)
1	10	32	3.2
2	8	16	2
3	3	8	2.66
4	1	4	4

In descending order based on $f(i)/s(i)$

i	s(i)	f(i)	f(i)/s(i)
4	1	4	4
1	10	32	3.2
3	3	8	2.6
2	8	16	2

Costs for above arrangements are as follows:

$$\begin{aligned}\text{cost(freq)} &= 32 * (10) + 16 * (10 + 8) + 8 * (10 + 8 + 3) + 4 * (10 + 8 + 3 + 1) \\ &= 320 + 288 + 168 + 88 \\ &= 864 \end{aligned} \tag{3}$$

$$\begin{aligned}\text{cost}(f(i)/s(i)) &= 4 * (1) + 32 * (1 + 10) + 8 * (1 + 10 + 3) + 16 * (1 + 10 + 3 + 8) \\ &= 4 + 352 + 112 + 352 \\ &= 820 \end{aligned} \tag{4}$$

After comparing (3) and (4), we can see that descending ordering based on ratio of frequency to size gives a better performance than increasing ordering based on frequency.

Hence this greedy approach is not optimal.

(c) Yes. This greedy algorithm gives the optimal solution.

π^a be the permutation where programs are in decreasing order of $f(i)/s(i)$.
And π^b be the permutation where programs yield the optimal solution i.e. the minimum cost.

Case 1: Permutation of programs in π^a is same as π^b
This implies that the greedy algorithm is optimal.

Case 2: Permutation of programs in π^a is different from that in π^b
This implies that there exist two programs P^a and P^b that have different indices in permutations π^a and π^b.

Lets say greedy ranks P^a before P^b.
$\implies$ $f(a)/s(a) \geq f(b)/s(b)$
$\implies$ $f(a) \geq f(b)$ or $s(a) \leq s(b)$ or both.
$\implies$ Greedy ranks the program P^a higher because it has higher frequency of use or smaller size or both.

This satisfying both the necessary criterion, greedy tries to minimize the cost locally which in turn results in minimum overall cost.

Thus we can say that there should not exist any programs P^a and P^b (with distinct $f(i)/s(i)$ value) which have different ranks in π^a and π^b.

This implies that the greedy algorithm is actually the optimum solution.