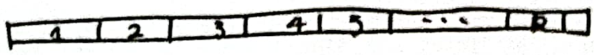
Jeff Erickson: Algorithms.

§4 Greedy Algorithm.

Eg 1. ~~Greed~~ Storing file on tape.

<u>SETUP</u> · ~~A tape~~ · Reading a tape: FastFwd all other files



If stored in order, the cost of getting kth file is

$$cost(k) = \sum_{i=1}^{k} L[i] \rightarrow \text{length of i-th file.}$$

If equally stored, total expected cost is

$$\mathbb{E}[cost] = \sum_{k=1}^{n} \frac{cost(k)}{n} = \frac{1}{n} = \sum_{k=1}^{n} \sum_{i=1}^{k} L[i].$$

Now, some files becomes more expensive to read, others are cheaper.

Let $\pi(i) :=$ index of file stored at position i on the tape

Then the expected ~~value~~ cost of the permutation π is

$$\mathbb{E}[Cost(\pi)] = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{k} L[\pi(i)].$$

Put cheapest to access first, then more expensive ones.

Lemma. $\mathbb{E}[cost(\pi)]$ is minimized when $L(\pi(i)) \le L(\pi(i+1))$ $\forall i$.

Proof. Suppose there is some i that $L[\pi(i)] > L[\pi(i+1)]$. Let $a := \pi(i)$ and $b := \pi(i+1)$. If we swap files a and b, the cost of accessing a increases by $L[b]$, and the cost of accessing b decreases by $L[a]$. This will affect the expected cost by $\frac{1}{n}(L[b]-L[a])$. However, $L[b]-L[a] < 0$, and it's an improvement. By induction, we can swap all possible vals like this getting an optimal situation. $\square$

Example 2. Like in Example 1, and we have an array $F[1..n]$. of access frequencies for each file.

- File i will be accessed <u>exactly</u> $F[i]$ times over the lifetime of the tape.

Total cost of accessing all files is

$$\sum cost(\pi) = \sum_{k=1}^{n}\left(F[\pi(k)]\sum_{i=1}^{k} L[\pi(i)]\right)$$

$$= \sum_{k=1}^{n}\sum_{i=1}^{k}\left(F[\pi(k)]\, L[\pi(i)]\right).$$

Minimize that.

Fix LENGTH : ~~Asce~~ Decending frequencies

Fix FREQ : Size increasing.

? Sort by ratio $\frac{L}{F}$?

Lemma. $\sum cost(\pi)$ is minimized when $\frac{L[\pi(i)]}{F[\pi(i)]} \le \frac{L[\pi(i+1)]}{F[\pi(i+1)]}$.

Proof. Suppose $\frac{L[\pi(i)]}{F[\pi(i)]} > \frac{L[\pi(i+1)]}{F[\pi(i+1)]}$ for some idx i, to simplify notation, let $a := \pi(i)$, $b := \pi(i+1)$.

If we swap files a and b, then the cost of accessing a increases by $L[b]$, and accessing b decr by $L[a]$. Overall, the swap change the total cost by $L[b]F[a] - L[a]F[b] \underset{\sim}{< 0}$ (hypothesis).

Thus, if out of order, improve total cost by swapping. $\boxed{2}$

Example 3. Scheduling Classes

<u>SETUP</u> Given 2 arrays $S[1..n]$ and $F[1..n]$

Start ends

for each i. $0 \le S[i] < F[i] \le M$, $\forall i$.

- Choose largest possible subset $X \in \{1, 2, \cdots, n\}$, s.t. for any pair $i, j \in X$, either $S[i] > F[j]$ or $S[j] > F[i]$ (no intersection).

Take class 1 or not?

▷ Recursive appoarch

Let $B :=$ set of classes ends before class 1 starts

$$\{i : 2 \le i \le n \land F[i] < S[1]\}$$

$A :=$ set of classes start after class 1 ends

$$\{i \mid 2 \le i \le n \land S[i] > F[1]\}.$$

Try these two until we can find an optimal schedule.

▷ Intunitive appoarch. first class to finish as early as possible.

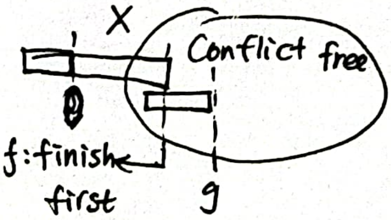
> Scan through classes in order of <u>finish time</u>
> whenever encounter a class <u>doesnt conflict with your latest class</u> so far, take it

```
GREEDY SCHEDULE (S[1..n], F[1..n])
    sort {S, F} w.r.t. finish time.
    count ← 1
    X[count] ← 1
    for i ← 2 to n
        if S[i] > F[X[count]]
            count ← count + 1
            X[count] ← i
    return X[1..count]
```

Lemma 3. At least 1 maximal conflict-free schedule includes ~~that~~ the class that finishes first.

Proof. Let f be class that finishes first.

Suppose we have a maximal conflict-free schedule X does not include f, Let g be the first class in X to finish.



Since f finishes before g does, f cannot conflict any classes in the set $X \setminus \{g\}$.

Thus, the schedule $X' = X \cup \{f\} \setminus \{g\}$ is also maximal.

Theorem 4. The greedy schedule is an optimal schedule.

Proof. Let $\langle g_1, g_2, \cdots, g_k \rangle$ be the seq of classes chosen by greedy algo, sorted by starting time, and we have a maximal conflict-free schedule

$$S = \langle g_1, g_2, \cdots, g_{j-1}, \underbrace{c_j, c_{j+1}, \cdots, c_m}_{\substack{\text{probably new.} \\ \text{don't conflict } g_1 \cdots g_{j-1}.}} \rangle,$$

sorted by starting time.

By construction, the jth greedy choice g_j does not conflict with any earlier class $g_1, g_2, \cdots, g_{j-1}$ and because our schedule S is conflict-free, neither does c_j.

Moreover, g_j has earliest finish time among all classes don't conflict with earlier classes.

In particular, g_j finishes before c_j. It follows that g_j does not conflict with any of the later classes $c_{j+1}, \cdots, c_m$. Thus, the modified $S' = \langle g_1, g_2, \cdots, g_{j-1}, \underline{g_j}, c_{j+1}, \cdots, c_m \rangle$

is also conflict-free.

→ By induction. $\square$

~~Sum~~ v

1. The general pattern.

- ASSUME there's optimal solution different from greedy solution.
- FIND <u>first</u> difference between two solutions.
- ARGUE we can change the optimal choice for the greedy choice without making the solutions worse. (might not make it better).

and by induction, we have

optimal solution $\ni$ greedy solution.