CS 4800: Algorithms & Data

Lecture 11

February 16, 2018

Comparing genomes

- Given 2 strings/genes
  - $X = x_1x_2 \ldots x_m$
  - $Y = y_1y_2 \ldots y_n$

- Find alignment of X and Y with min cost
  - Each position in X or Y that is not matched cost 1
  - For each pair of letters p, q, matching p and q incurs mismatch cost of $a_{p,q}$

S	T	E	P	-
-	T	O	-	S
Cost 1		Cost $a_{e,o}$	Cost 1	Cost 1

Subproblems

- Best(i, j): minimum alignment cost for 2 strings x_i, …, x_m and y_j, …, y_n

Guess to align x[i:] and y[j:]

x_i	x_{i+1}	…	x_{m-1}	x_m
y_j	y_{j+1}	…	y_{n-1}	y_n

- How to align first characters?
- 3 possibilities:
  - Match x_i and y_j
  - x_i not matched
  - y_j not matched

Recursive relation

- $Best(i,j) = \min \begin{cases} a_{x_i,y_j} + Best(i+1, j+1) \\ 1 + Best(i+1, j) \\ 1 + Best(i, j+1) \end{cases}$
- Evaluation order: from large i to small i, from large j to small j

Whole algorithm

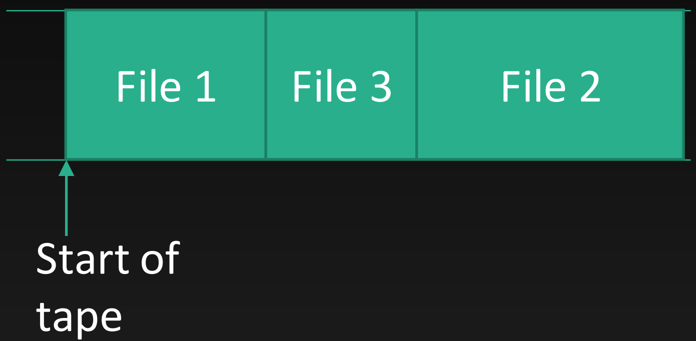
- Initialize
  - $Best(m+1, n+1) \ = \ 0$ // aligning 2 empty strings
  - $Best(m+1, j) = n - j + 1$ for j from 1 to n
  - $Best(i, n+1) = m - i + 1$ for i from 1 to m
- For i from m down to 1
  - For j from n down to 1
    - $Best(i, j) \ = \ \min \begin{cases} a_{x_i, y_j} + Best(i+1, j+1) \\ 1 + Best(i+1, j) \\ 1 + Best(i, j+1) \end{cases}$
- Return $Best(1,1)$

Greedy algorithms

Files on tape

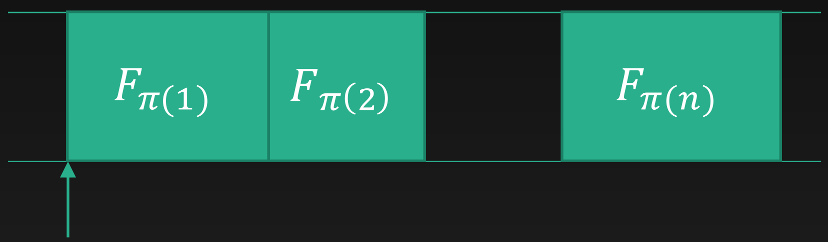
Tape storage

- n files of lengths $L_1, L_2, \ldots, L_n$
- To access a file on tape, need to scan pass all previous files
- Want an ordering to store the files to minimize then time to access a random file



Precise objective

- Say the file are stored according to permutation π



- Time to access the i-th file is $\sum_{j=1}^{i} L_{\pi(j)}$
- Expected accessing time of a random file is

$$\mathrm{cost}(\pi) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{i} L_{\pi(j)}$$

Example

File 1	File 2	File 3
10	5	15

- Time to access file 1: 10
- Time to access file 2: 15
- Time to access file 3: 30
- Expected accessing time: $\frac{1}{3}(10 + 15 + 30) = 18.33$

Better ordering

- Swap files 1 and 2
- Time to access file 2: 5
- Time to access file 1: 15
- Time to access file 3: 30
- Expected accessing time: $\frac{1}{3}(5 + 15 + 30) = 16.67$

Greedy strategy

- Order the files in non-decreasing sizes

Exchange argument

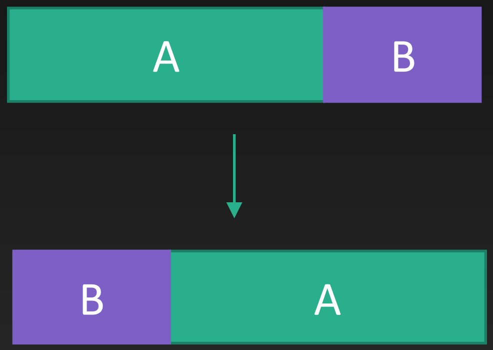
Claim. $\text{cost}(\pi)$ is minimized when $L_a \leq L_b$ for all pairs of consecutive files a and b in the ordering.

Proof.

Suppose $L_a > L_b$ for some consecutive files *a* followed by *b*.

If swap *a* and *b*,

- Cost of accessing *a* increase by L_b
- Cost of accessing *b* decrease by L_a



Overall, average accessing cost change by $(L_b - L_a)/n$

$L_b < L_a$ so the average accessing cost decreases.

Thus, can improve accessing time whenever there is a consecutive pair with decreasing sizes

Non-uniform frequencies

- File i is accessed F_i times
- Want to minimize total access time

$$\mathrm{cost}(\pi) = \sum_{i=1}^{n} \left(F_{\pi(i)} \sum_{j=1}^{i} L_{\pi(j)} \right)$$

Example

File 1 size: 5 freq: 2	File 2 size: 2 freq: 1	File 3 size: 8 freq: 5

- Time to access file 1: 5
- Time to access file 2: 7
- Time to access file 3: 15
- Total accessing time: $2 \cdot 5 + 1 \cdot 7 + 5 \cdot 15 = 92$

Better ordering

File 3	File 2	File 1
size: 8	size: 2	size: 5
freq: 5	freq: 1	freq: 2

- Time to access file 3: 8
- Time to access file 2: 10
- Time to access file 1: 15
- Total accessing time: $5 \cdot 8 + 1 \cdot 10 + 2 \cdot 15 = 80$

Greedy algorithm

- Sort the files by the ratio Length/Freq.

Exchange argument

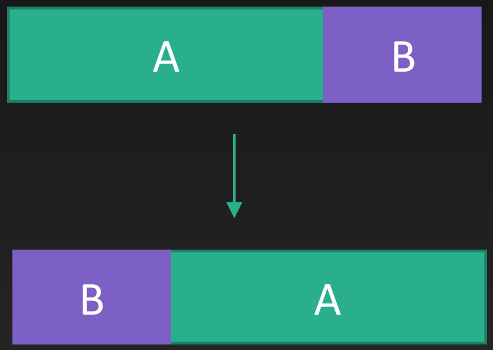
Claim. $\mathrm{cost}(\pi)$ is minimized when $L_a/F_a \leq L_b/F_b$ for all consecutive pair of files a followed by b.

Proof.

Suppose $\frac{L_a}{F_a} > \frac{L_b}{F_b}$ for some consecutive files a followed by b.

If swap a and b,

- Cost of accessing a increase by L_b
- Cost of accessing b decrease by L_a



Overall, average accessing cost change by

$$L_b F_a - L_a F_b$$

$\frac{L_a}{F_a} > \frac{L_b}{F_b}$ so the average accessing cost decreases.

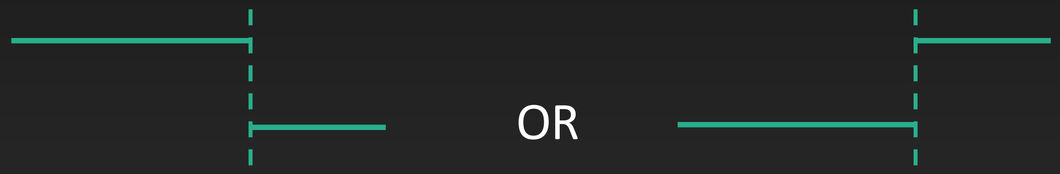
Thus, can improve accessing time whenever there is an out of order pair.

Scheduling

Movie	Start	End
Blair Witch	10:30	12:00
Bridget Jones’s Baby	10:45	12:45
Deepwater Horizon	10:15	12:10
Masterminds	12:30	2:00
Miss Peregrine’s	1:15	3:20

Problem statement

- n activities
- Start times : $s_1, s_2, \ldots, s_n$
- Finish times: $f_1, f_2, \ldots, f_n$
- Find largest subset of activities that are compatible

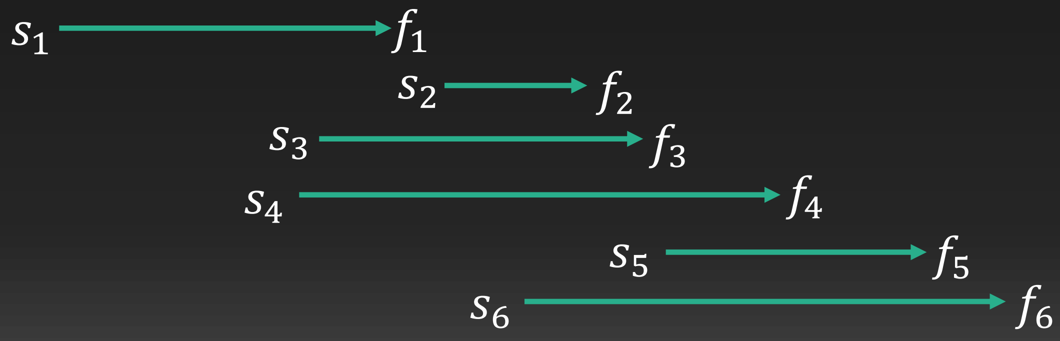


$f_i \leq s_j$

$f_j \leq s_i$

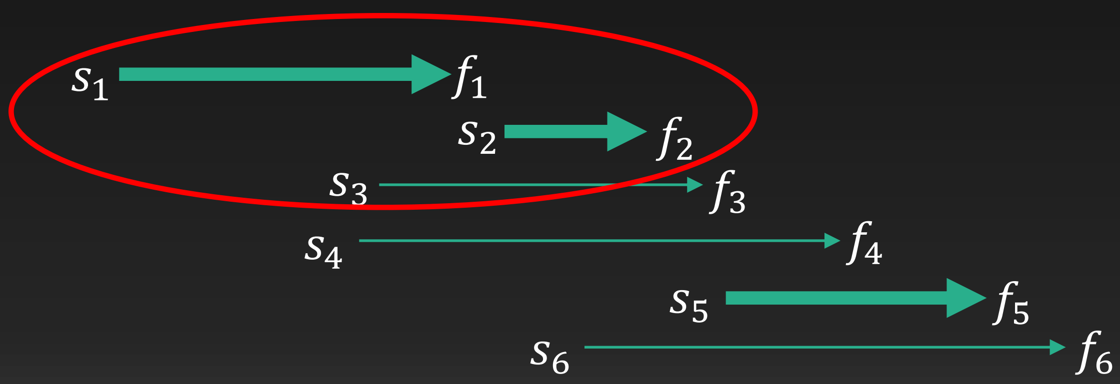
Problem statement

- n activities
- Start times : $s_1, s_2, \ldots, s_n$
- Finish times: $f_1 \leq f_2 \leq \cdots \leq f_n$ (sorted)
- Find largest subset of activities that are compatible



Dynamic Programming

- Best(i): Maximum # compatible activities finishing by f_i
- Optimal substructure: consider activities comprising Best(i) (e.g. best(5) is {1,2,5})



- Claim. The prefix (e.g. {1,2}) is optimal choice if restricted to activities finishing before the start of last activity (s_5).