Greedy Algorithms

Optimize

Solvable by local choices
without "lookahead"

e.g.

$[1, 7, 20, 13, 12, 5, 3, 2]$

Goal: Pick a subset S maximizing $\sum_{x \in S} x$.

$|S| = 3$

Algo: Take 3 biggest.

Proof of correctness: By contradiction.

Let S be the 3 biggest. Suppose $\sum_{x \in S} x$ is not maximal. So there is a subset T of size 3 w/ $\sum_{x \in T} x$ maximal.

Consider $T \setminus S = \{x \in T : x \notin S\}$. Pick $x_0 \in T \setminus S$. By construction of S, $x_0 \leq \min\{x \in S\}$. Consider what happens if we replace one element of S w/ x_0. (Argue by cases) $\Rightarrow$ sum can't go up.

Keep replacing until $S' = T$.

Each replacement didn't increase the sum, and @ the end the sum is maximal. Since sum wasn't increasing, $\sum_{x \in S} x$ must have already been maximal. □

(val, wt) and W - max wt.

[(1, 10), (100, 1), (1000, 100), ...]

Goal: Pick a subset S maximizing $\sum_{x \in S} val(x)$
Subject to $\sum_{x \in S} wt(x) \leq W$

Files File i length L(i)

Put files in same order on the tape &

Goal: Min. expected time to access the j^{th} file
(pick j randomly)

Suggestion: Sort them, smallest first.

Pick j randomly

$$E[\text{time}] = \frac{1}{n} \sum_{j=1}^{n} \text{cost to access } j^{th} \text{ file}$$

$$= \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{j-1} L(i)$$

Claim: E[cost] is minimized when files are sorted by length (shortest first

By contradiction. Suppose not. Files are in some unsorted order which (supposedly) minimizes E[cost].

$\exists\ i, j \quad i < j \quad L(i) > L(j)$

Consider first file out of order

$L(i) > L(i+1)$

What happens to E[cost] if we swap them?

$$\frac{1}{n}\left[\sum_{j=1}^{i-1} \text{cost to access } j + \text{cost to access } i + \text{cost to access } i+1 + \sum_{j=i+2} \cdots j\right]$$

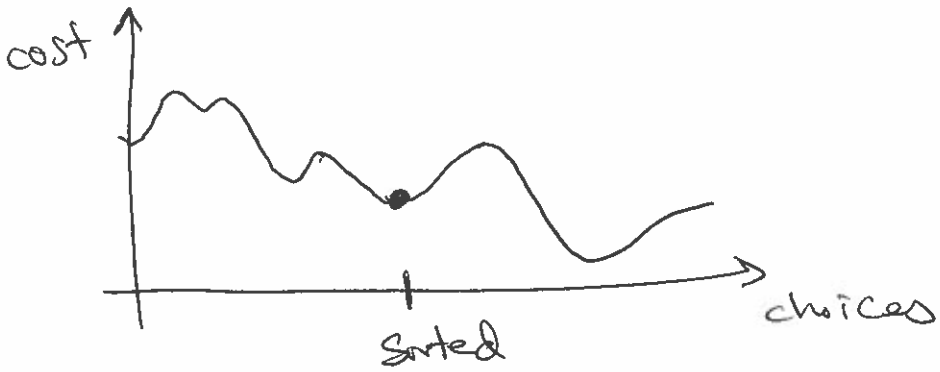
↑ does it change

cost to access i	before swap	after swap
cost to access i	$\sum_{k=1}^{i-1} L(k)$	$\sum_{k=1}^{i-1} L(k) + L(i+1)$
... i+1	$\sum_{k=1}^{i} L(k)$	$\sum_{k=1}^{i-1} L(k)$

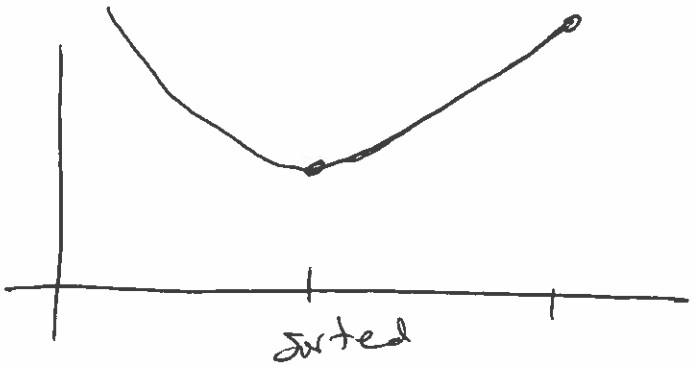
after swap − before swap = $L(i+1) - L(i) < 0$.

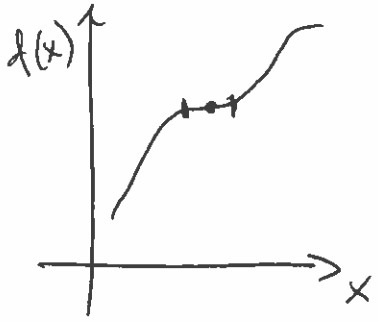
Each swap which puts ~~file~~ K makes the list "more sorted" decreases cost.

Do all nec. swaps to sort. Any additional swap will incr. cost.



local optimum





Finding opt value $\neq$ being fastest algorithm.

Find min $y = x^2 - 1$

25c 20c 10c 5c

Make change for 40c (min # coins used)

Huffman Coding

Alphabet $a, \ldots, z$

Encode as bit-strings $\text{code}(a), \text{code}(b) \ldots$

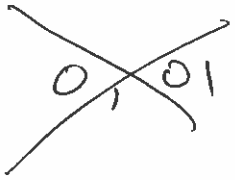
Frequences $p(a), p(b) \ldots, p(z)$

Expected codelength $\sum_{i \in \{a,\ldots,z\}} p(i) \cdot \text{len}(\text{code}(i))$

Goal: Construct prefix-free code minimizing expected code length

How to delimit indiv. chars?

Prefix(-free) code: no codeword is a prefix of any other codeword

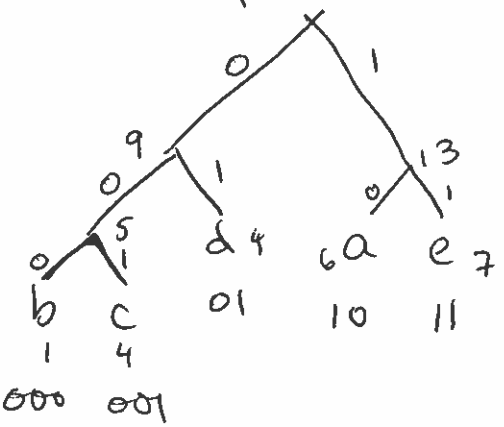


0, 100, 1010, 1011 ✓

0|100|100|1010|1011 1010

Ex

a	b	c	d	e
6	1	4	4	7



Error