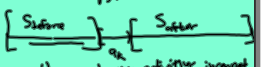
Greedy Algos

- Optimization problems
- Optimal substructure
- Recursive solution
- Dynamic Programming sol'n?
- Greedy solution

Greedy algo:
Takes orig problem and breaks it down into one or more subproblems that looks the best

---

Set of activities $\{a_1, \ldots, a_n\}$
each a_k has a
start time s_k
finish time f_k
duration of a_k is the interval $[s_k, f_k)$
Find biggest subset of activities that are compatible (durations don't overlap).



- throw out any activities incompat with a_k
- two subproblems of remaining activities:

$S_{before} = \{a_j : f_j \le s_k\}$

$S_{after} = \{a_j : f_k \le s_j\}$

only subproblems of this type: $S_{ij} = \{a_k : f_i \le s_k \text{ & } f_k \le s_j\}$

all activities between a_i and a_j
Add two dummy activities:
$a_0 = [-\infty, \infty)$
$a_{n+1} = [\infty, \infty)$
Solve for S_{ij} $\{0 \le i, j \le n+1\}$

Greedy algorithm:
Sort all activities by finish time. Re-index so that $f_1 \le f_2 \le f_3 \le \cdots \le f_n$
While there are activities left
- choose activity with earliest ~~start~~ finish time
- transfer it to the solution
- remove all activities incompatible with this choice.

$a_1 \longrightarrow$ solution
remove all a_i s.t. $s_i < f_1$ (incompatible with a_1)

Don't need a table of sol'ns to sub problems. Usually a single recursive alg. is fine.

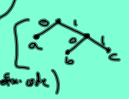
Huffman coding

Text file of chars from some alphabet $C = \{s_1, s_2, \ldots, s_n\}$
Encode in binary as efficiently (space) as possible
(optimal compression)
assuming that each symbol is encoded by a binary string.

Prefix code:
$s_1 \longrightarrow b_1$
$s_2 \longrightarrow b_2$
$\vdots$
$s_k \longrightarrow b_k$
$b_1, \ldots, b_k \in \{0,1\}^*$ codewords
such that no b_i is a prefix of b_j for any $i \ne j$.

a → 0
b → 10
c → 11
11001011110 ... = c a a b a c b ...

Binary coding tree (gives a prefix code)

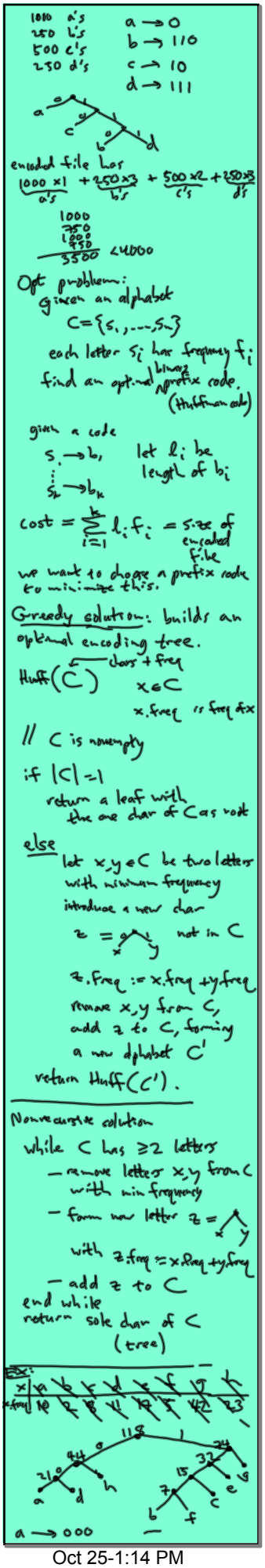


$C = \{a, b, c, d\}$
file F has 1000 a's
250 b's
500 c's
250 d's

a → 00
b → 01
c → 10
d → 11

Encoded file has 4000 bits

Oct 25-12:28 PM


1000 a's
250 b's
500 c's
250 d's
a → 0
b → 110
c → 10
d → 111
a
c
b
d
encoded file has
1000 ×1 + 250×3 + 500×2 + 250×3
a's b's c's d's
1000
750
1000
750
3500 < 4000
Opt problem:
Given an alphabet
C={s1,...,sn}
each letter si has frequency fi
find an optimal binary prefix code.
(Huffman code)
given a code
s1 → b1
sk → bk
let li be length of bi
cost = Σ li fi = size of encoded file
we want to choose a prefix code to minimize this.
Greedy solution: builds an optimal encoding tree.
Huff(C) chars + freq
x∈C
x.freq is freq of x
// C is nonempty
if |C|=1
return a leaf with the one char of C as root
else
let x,y∈C be two letters with minimum frequency
introduce a new char
z =
not in C
z.freq := x.freq + y.freq
remove x,y from C,
add z to C, forming a new alphabet C'
return Huff(C').
Nonrecursive solution
while C has ≥2 letters
– remove letters x,y from C with min frequency
– form new letter z =
with z.freq := x.freq + y.freq
– add z to C
end while
return sole char of C
(tree)
Ex:
x
x.freq
a
b
c
d
e
f
g
h
10
2
8
11
17
5
42
23
118
44
74
37
21
15
7
a
d
h
e
g
b
f
c
a → 000

Oct 25-1:14 PM