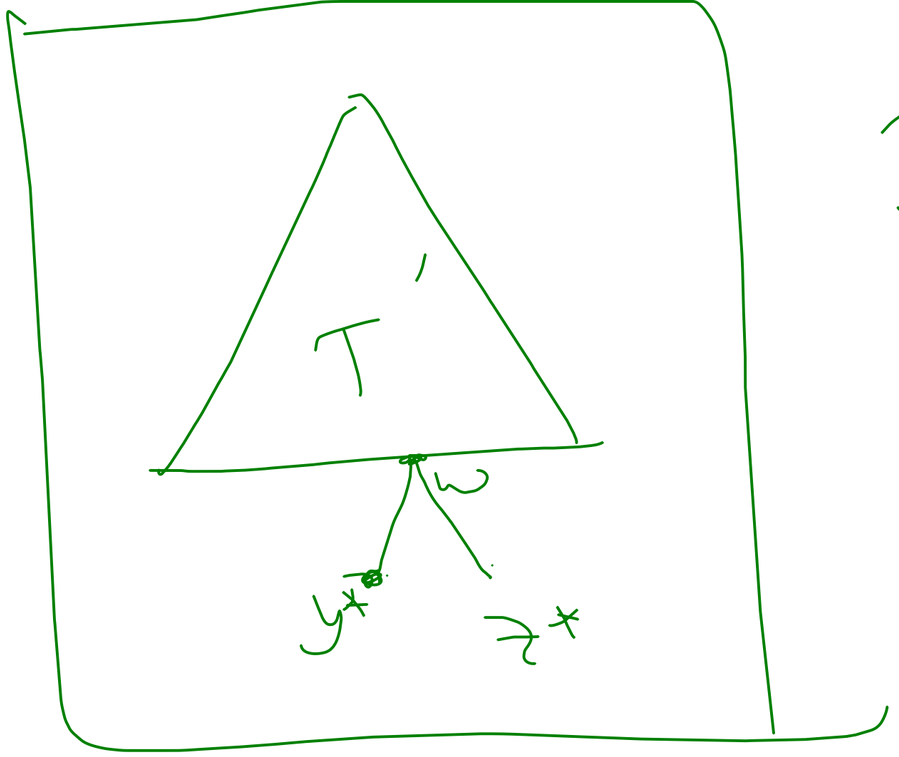
Huffman's Algorithm

Input A alphabet S with given freq's $\{f_x : x \in S\}$

Output A tree T that represents an optimal prefix code for S.

1. If S contains two letters a and b
   then encode a with a & b with 1.
2. Else Let y^* & z^* be the 2 lowest
   frequency letters in S.
3. Form a new alphabet S' by deleting
   y^* & z^* and adding a new letter w
   of frequency $f_w = f_{y^*} + f_{z^*}$
4. Recursively construct a prefix code
   for S' with tree T'.



Define a prefix code
Z for S as follows:
Start with T'
Take the leaf node
labeled w & add
two children below it
Label them with y^*, z^*

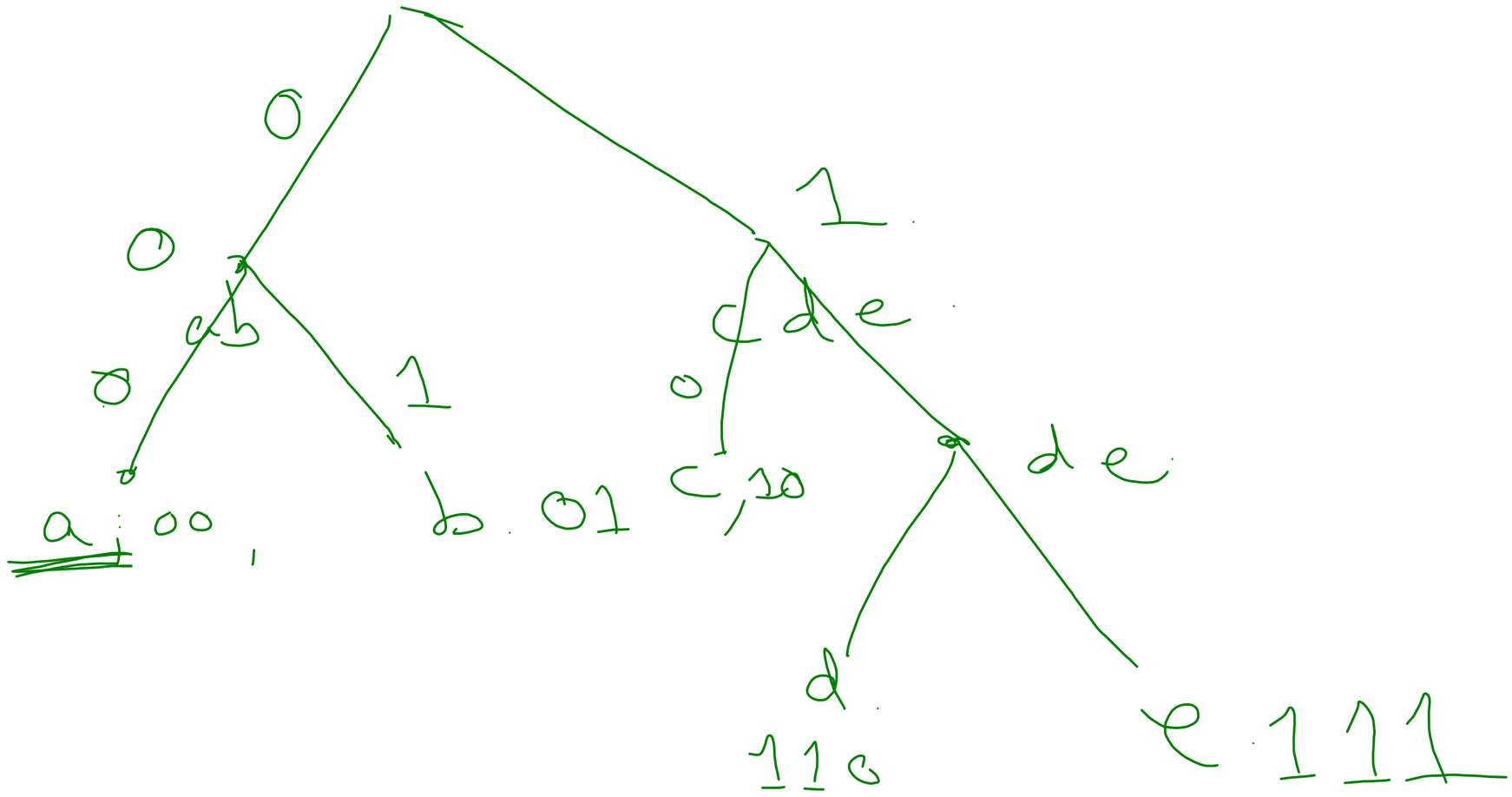
Ex $S = \{a, b, c, d, e\}$

$f_a = .32$, $f_b = .25$, $f_c = .20$, $f_d = .18$, $f_e = .05$

1 $S^1 = \{a, b, c, de\}$ $f_a = .32$, $f_b = .25$, $f_c = .20$, $f_{de} = .23$

2 $S^{11} = \{a, b, cde\}$ $f_a = .32$, $f_b = .25$, $f_{cde} = .43$

3 $S^{111} = \{ab, cde\}$ $f_{ab} = .57$, $f_{cde} = .43$


0
1
0
0
1
0
a : 00
b 01
c 10
d
e
11 0
1 1 1

Thm Huffman code achieves the minimum average no. of bits per letter of all prefix codes on S.

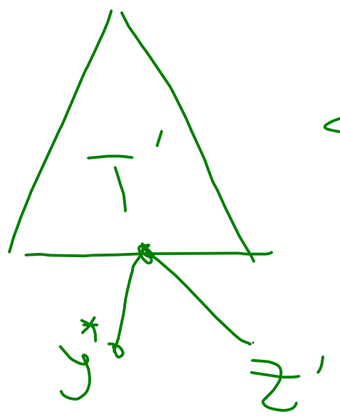
Pf We shall prove by induction on |S|.

If |S|=2 then nothing to prove.

So assume that Huffman's alg. gives an optimal prefix code on

On alphabet of size 12-1 & let S be an alphabet of size k.

Let y^*, z^* be the lowest frequency letters of S.



S' into a single letter w & we then construct T' for S'. T is then obtained from T' by adding 2 children below w & labelling them by y^*, z^*

Claim $ABL(T') = ABL(T) - fw$.

Observe that for any letter x after y^*, z^*, the depth of x is the same in both T & T'. For each of y^*, z^* the depth in T is 1 more than the depth of w in T'.

Hence

$$ABL(T) = f_{y^*}\, depth_T(y^*) + f_{z^*}\, depth_T(z^*) + \sum_{x \neq y^*, z} f_x\, depth_T(x)$$

$$= (f_{y^*} + f_{z^*})\left(1 + depth_{T'}(w)\right) + \sum_{x \neq y^*, z} f_x\, depth_{T'}(x)$$

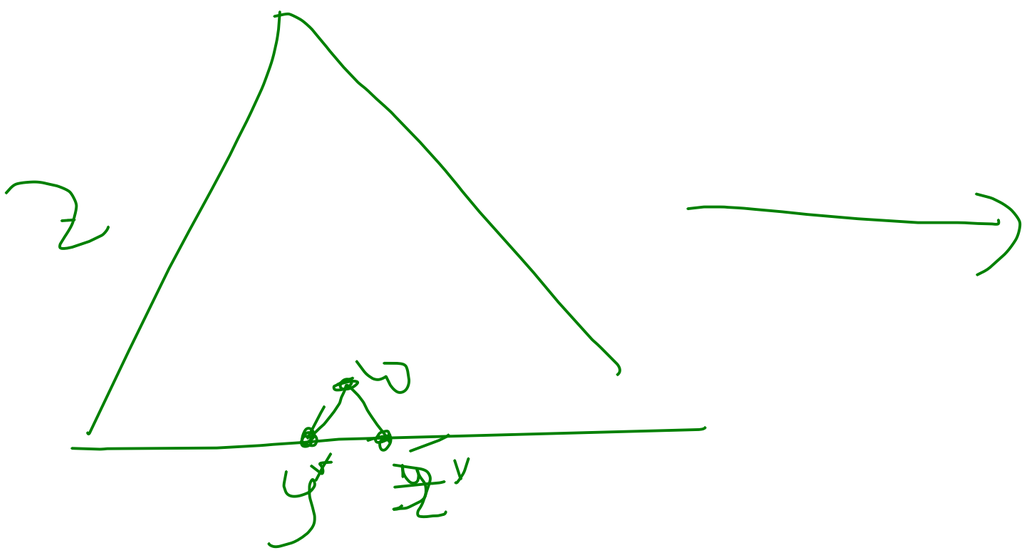
$$= f_w + f_w\, depth_{T'}(w) + \sum_{x \neq y^*, z^*} f_x\, depth_{T'}(x)$$

$$= f_w + ABL(T')$$

Now assume that the tree given by Huffman's algorithm is not optimal

This mean there is a labeled tree T s.t $ABL(T) < ABL(T)$.

On T, y^*, z^* are labels of leaves that are siblings



Construct T' from T in the same way Z' is obtained from Z.

Hence $ABL(Z') = ABL(Z) - f_w$.

$ABL(Z) < ABL(T)$

$ABL(T') = ABL(Z') - f_w < ABL(T) - f_w = ABL(T')$

Fractional Knapsack

Given K & n items of wts $w_1, \ldots, w_n$ & values $v_1, v_2, \ldots, v_n$, find x_i, $0 \le x_i \le 1$ s.t. $\sum_{i=1}^{n} x_i w_i \le K$ and s.t. the following exp is maximized.

$$\sum_{i=1}^{n} x_i v_i$$

Greedy Algorithm for fractional knapsack

- Calculate the value per unit $\rho_i = \frac{v_i}{w_i}$
- Sort the item by decreasing ρ_i
- Let the sorted item sequence by $1, 2, \ldots, n$
  & for each i the corr^n value-per-unit is ρ_i & wt w_i
- Let K be the current wt limit. Initially $K = K$
  On each iteration we choose item i from the head of the unselected list
  - if $K \geq w_i$ then $x_i = 1$ (item i is chosen)
  - if $K < w_i$, set $x_i = K/w_i$ (the fraction K/w_i of item i is chosen) & then halt

Thm The greedy algorithm gives an optimal soln in time $O(n \log n)$

Pf Let the sorted items be $1, 2, \ldots, n$ so that $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$.

Let $X = (x_1, x_2, \ldots, x_n)$ be the soln given by our algorithm.

Let $O = (y_1, \ldots, y_n)$ be an optimal soln **

If $X = O$, then X is optimal.

So assume $X \neq O$

Let i be the smallest index s.t $x_i \neq y_i$

(clearly, $x_i > y_i$ (why?)

x_1 x_2 ... x_{i-1} x_i

y_1 y_2 ... y_{i-1} y_i

since greedy alg takes max possible

Let $\alpha = x_i - y_i$

Note that the wt of item i in X exceeds that of O by αw_i

** In both X & O the wts are full

ie $\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K$

Let O' be constructed from O as follows. Set $y'_j = y_j = x_j$ for $j < i$

Set $y'_i = x_i$

On O from items $i+1, \ldots, n$ subtract weights $\varepsilon_{i+1}, \varepsilon_{i+2}, \ldots, \varepsilon_n$ with total sum $\boxed{\sum w_i}$

The decrease in value in O'

$$\leq \varepsilon_{r+1}\rho_{r+1} + \varepsilon_{r+2}\rho_{r+2} - - - + \varepsilon_n \rho_n$$

$$\leq \left(\varepsilon_{r+1} + - - + \varepsilon_n\right)\rho_r = r\, w_r \frac{v_r}{w_r} = \boxed{r v_r}$$

Hence the total value in O' is greater or equal to the total value in O. Since O is optimal, Value in O' = Value in O. Hence O' is optimal.

Proceeding in this manner

O can be converted to X.

Hence X is obtained.

Single Source shortest paths

Defn Given a directed graph $G=(V,E)$ & a wt. fn $w: E \to \mathbb{R}$, the wt or length of a path in G is the sum of the wts of the edges of the path. The single source shortest path problem is to find, for each $v \in V$, the smallest wt path from the ~~source~~ source s to v i.e. to find the shortest path from s to v.

We define

$$\delta(s,v) = \begin{cases} \text{the length of the shortest path from } s \text{ to } v & \text{if } v \text{ is reachable from } s \\ +\infty & \text{o.w.} \end{cases}$$

Dijkstra's Algorithm

Input A directed graph $G=(V,E)$, a source $s \in V$
& wt fn $w: E \to \mathbb{R}^+$. We take $w(v_i,v_j) = \infty$
if $(v_i,v_j) \notin E$, $v_i \neq v_j$, $w(v,v) = 0$.

Output For each $v \in V$, the minimum over all paths from s to v of the sum of wts of the edges of p.

Method: We construct a set $S \subseteq V$ s.t. the shortest path from s to each $v \in S$ lies wholly in S. The array $D[v]$ contains the wt of the current shortest path from s to v passing entirely through S.

1 $S \leftarrow \{s\}$

2 $D[s] = 0$

3 for each $v \in V - \{s\}$ do $D[v] = w(s,v)$

4 while $S \neq V$

begin

5 choose a vertex $z \in V - S$ s.t. $D[z]$ is min

6 Add z to S

7 for each $v \in V - S$ do

8 $D[v] = \min\{D[v], D[z] + w(z,v)\}$