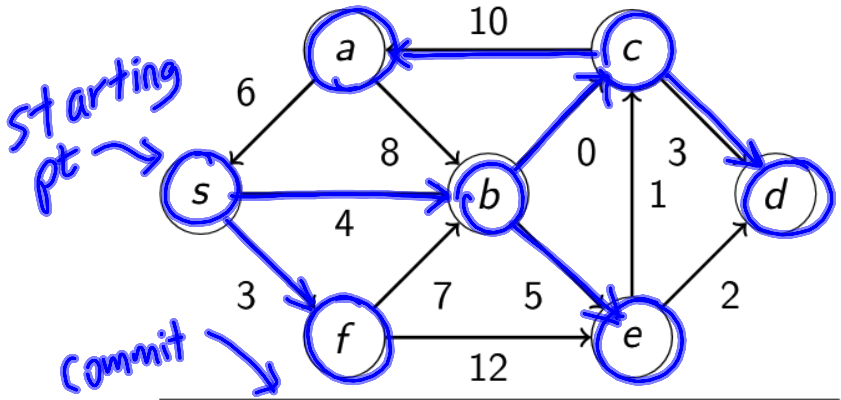
- Turn in homework to front of classroom.
- Be sure your name is on your homework.
- Be sure to *staple* if you have multiple pages.
  - A paper clip is not a staple.
  - A fold is not a staple.

Dijkstra's Algorithm



v	intree(v)	parent(v)	dist(v)
s	~~F~~ T	N/A	0
a	~~F~~ T	c	$\cancel{\infty}$ 14
b	~~F~~ T	s	$\cancel{\infty}$ 4
c	~~F~~ T	b	$\cancel{\infty}$ 4
d	~~F~~ T	c	$\cancel{\infty}$ 7
e	~~F~~ T	~~f~~ b	$\cancel{\infty}$ ~~15~~ 9
f	~~F~~ T	s	$\cancel{\infty}$ 3

But why is it right?

Dijkstra's Algorithm

Running time?

$n = |V|$
$m = |E|$

with binary heap
(better possible w/ fib. heap)

for each vertex v **do** } $O(n)$
  intree(v) = false
  parent(v) = N/A
  dist(v) = ∞
dist(s) = 0 } $O(1)$
while $\exists$ vertex u with intree(u) = false **do** } $O(n \log n)$
  $O(\log n)$ → $u \leftarrow$ vertex with intree(u) = false and *smallest* dist(u) ← remove top of heap $O(\log n)$
  intree(u) = true
  once per edge total → **for** each vertex $v \in$ adj[u] **do**
    $O(\log n)$ { **if** d(v) > d(u) + w(u, v) **then**
      d(v) = d(u) + w(u, v) // add to heap or update
      parent(v) = u

removed elt not in free until

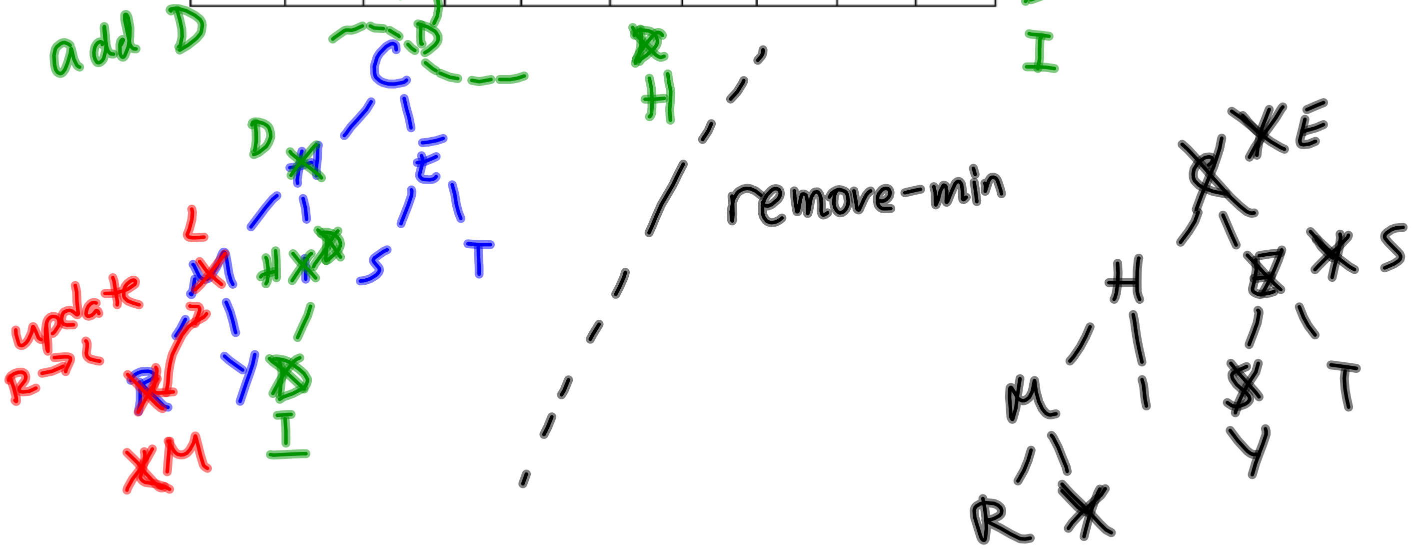
key: d(v)
value: v
$O(\log n)$

$O(m \log n)$ total

Total: $O((m+n) \log n)$

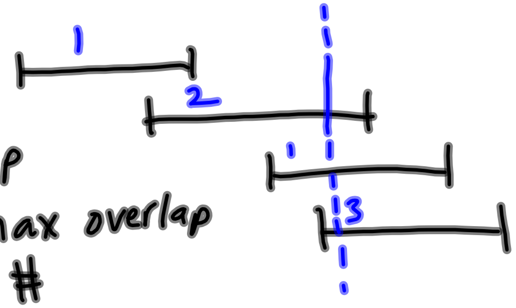
Heaps

i	1	2	3	4	5	6	7	8	9	10
H_i	C	~~H~~	E	M	~~X~~	S	T	R	Y	~~D~~



In-Class Exercise

Alg: give smallest unused key on arrival



max # keys = max overlap
and ≥ max overlap
in #

No alg could use fewer on any input.

Valid: no overlap w/ same room #

optimal/correct: no valid alt is better on # rooms

$B = \emptyset$ next=1

Create 2n "events":

- n arrival of group i
- n departure of group i

add all to a heap. H $O(n \log n)$

$O(n \log n)$ while $H \neq \emptyset$

$O(\log n)$ $e \leftarrow$ take event from H

$O(1)$ if e departure

add e's room # to B

else

if $B \neq \emptyset$, group gets arbit. # from B

else

group gets room # next.

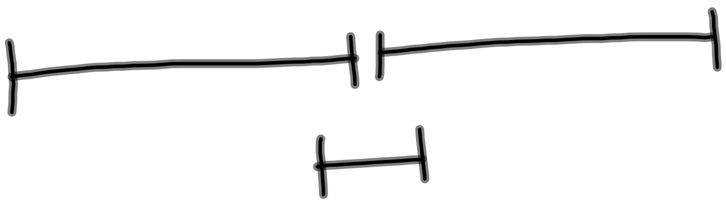
next ++

Possible Algorithm 1: First Starting Class

alg: 1 interval

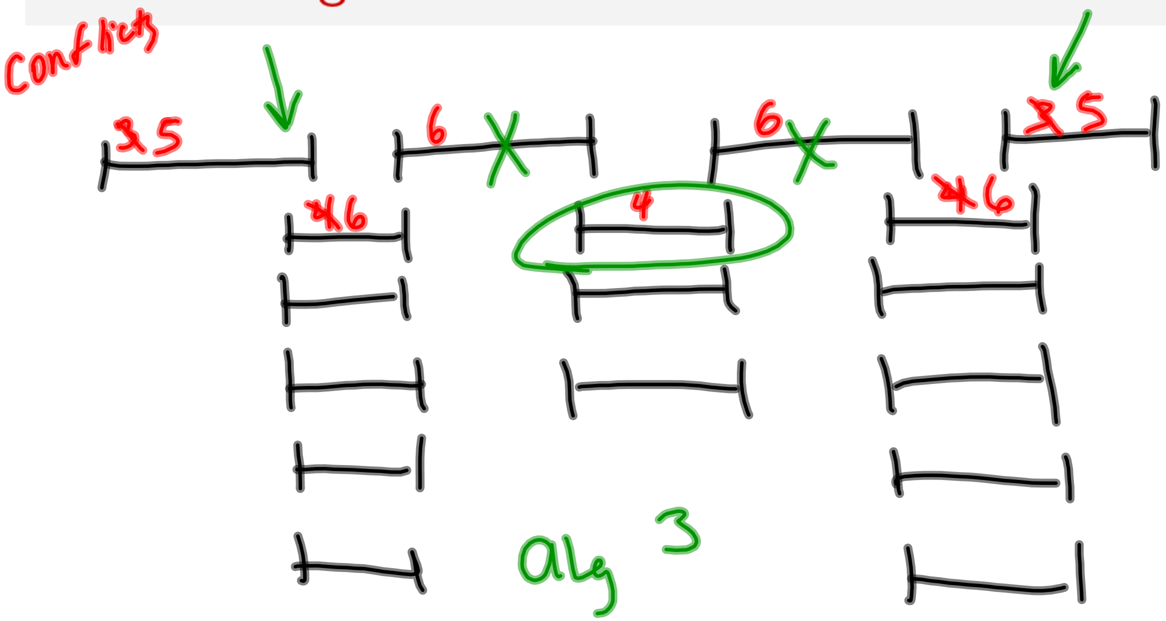
OPT: 4 intervals

Possible Algorithm 2: Shortest Class



alg: 1
OPT: 2

Possible Algorithm 3: Class With Fewest Conflicts



alg 3

OPT: 4

Possible Algorithm 4: First Ending Class

Proof Strategy:

- Some optimal uses same 1st

any OPT can be described:

- take 1st decision & any optimal
- remove it and any overlap
- recurse w/ Some optimal

Proving Correctness

Claim: There is an optimal solution that includes the first interval that we choose.

Note that I am not claiming that all optimal solutions do.

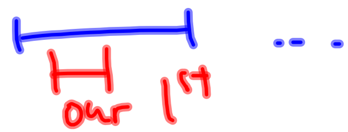
Let X be an optimal sol'n that doesn't include our 1st interval

$X' = X -$ X's first interval $+$ our 1st interval.

is X' even valid? Nothing in $X \cap X'$ overlaps.

does X' overlap our 1st?

X pre change:



our 1st ends $\leq$ X's first ends

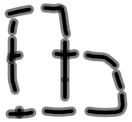
so no overlap b/c all others start after X's first ends

$|X| == |X'|$, so X' optimal size and valid

Would any optimal solution that includes our first interval also include any intervals that overlap with it?

What is left to do to prove that the rest of our algorithm is correct?

Induction as a recursive proof

Prove: any $2^n \times 2^n$ checkerboard w/ one square removed can be tiled w/  pieces

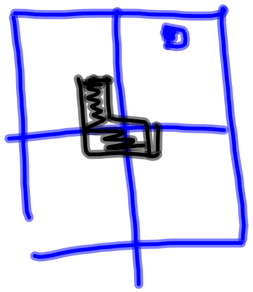
alg: $n=0$, trivial / done

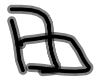
else: divide into four quadrants, $2^{n-1} \times 2^{n-1}$

One has a missing piece

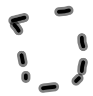
wlog say top right

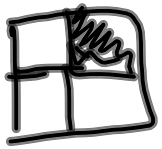
so call recursively on top right

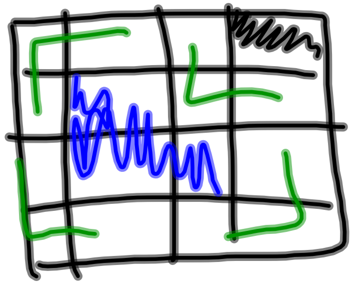


then place  as shown

and call 3 recursively

$n=0$: 

$n=1$:  and isomorphic

$n=2$ 

In-Class Exercise I

Give an algorithm that will achieve this goal with the fewest possible pizza parlors placed, please.

Go to 1st house + 5 miles,
place pizza parlor,
remove covered houses,
repeat

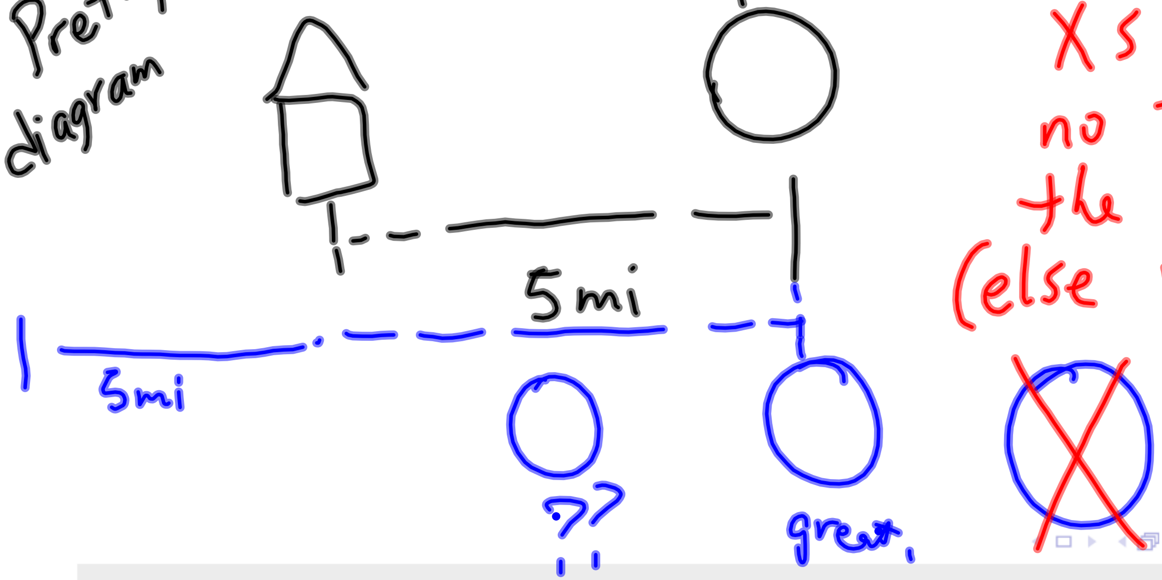
In-Class Exercise I

X ← arbitrary optimal placement

Prove that your algorithm achieves the optimal solution

Claim: Some optimal soln will place 1st pizza parlor 5 mi after 1st house

Pretty diagram



X's first is no further down the road than mine (else invalid soln)

What if X's first is before mine?

Let CX = houses covered by X's first

Let CM = houses covered by my 1st

subset

$CX \subseteq CM$

So this is a valid sol'n:

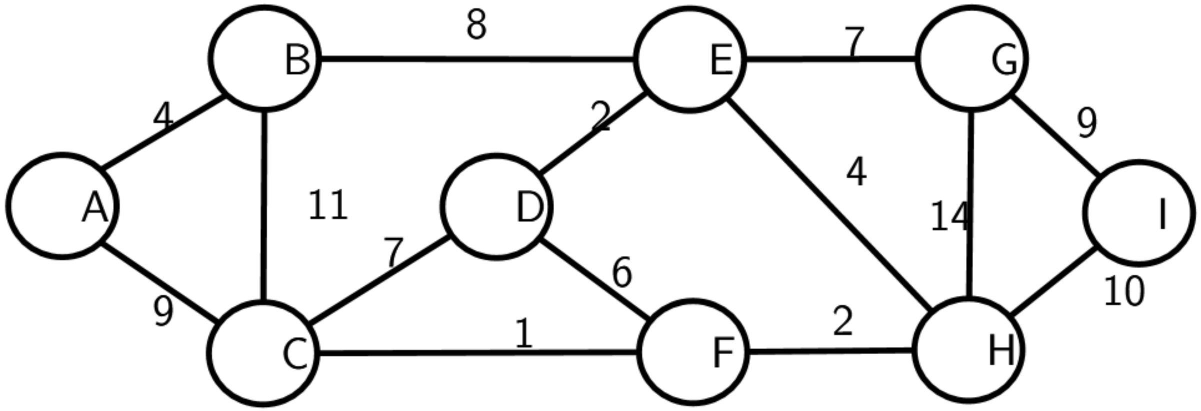
$X' = X -$ X's 1st + my first

$|X'| = |X|$ so X' same size as an optimal and valid,

so X' is an optimal

Minimum Spanning Trees

Read the problem description in your handout. Which edges would you keep for the following graph?



Reverse-Delete Algorithm

Could any valid solution contain a cycle?

Reverse-Delete Algorithm

Suppose C is a cycle within G. At least one edge in C won't be in our solution. Which edge and why?

The Reverse-Delete Algorithm

while G contains a cycle **do**
  Let C be a cycle within G
  Let e be a maximal edge within C
  Remove e from G
return G

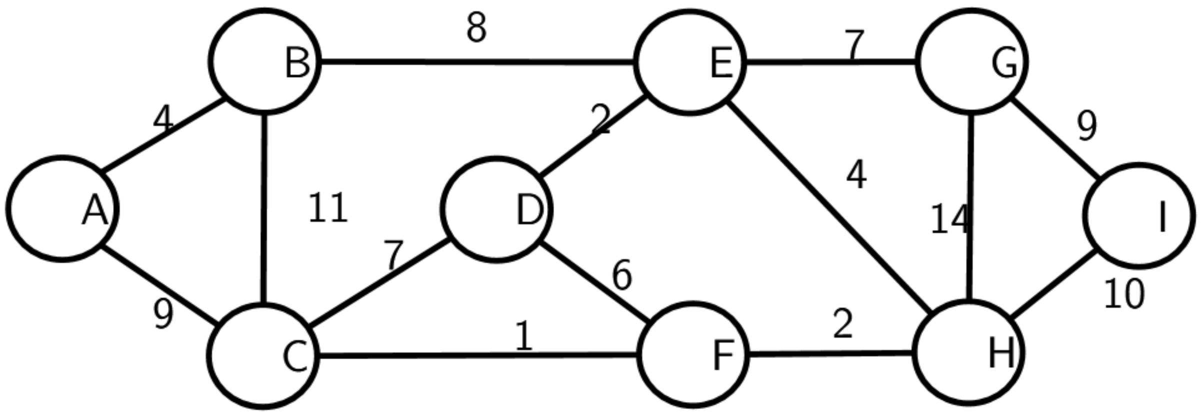
- Why is this correct?

The Cut Property

Modify Dijkstra's Algorithm?

for each vertex v **do**
  intree(v) = false
  parent(v) = N/A
  dist(v) = ∞
dist(s) = 0
while $\exists$ vertex u with intree(u) = false **do**
  $u \leftarrow$ vertex with intree(u) = false and *smallest* dist(u)
  intree(u) = true
  for each vertex $v \in$ adj[u] **do**
    if d(v) > d(u) + w(u, v) **then**
      d(v) = d(u) + w(u, v)
      parent(v) = u

Illustrate Prim/Jarnik/Dijkstra



Understanding the Problem

Example 1: What is the optimal schedule for the following?

Time	1	2	3
Deadline	2	4	6

Understanding the Problem

Example 2: What is the optimal schedule for the following?

Time	1	2	3	4
Deadline	2	4	6	6

Possible Algorithm 1

Sort the jobs by increasing time t_i; schedule them in that order.

Possible Algorithm 2

Sort the jobs by $d_i - t_i$; schedule them in that order.

Possible Algorithm 3

Sort the jobs by deadline d_i; schedule them in that order.

Proof

When deciding start times, don't leave any gaps; $s_{i+1} = s_i + t_i$.

Proof

Any schedule that doesn't agree with our algorithm has at least one pair of *consecutive* intervals $i, i+1$ that are *inverted* relative to our order.

Note: You may take this fact as a given for the related homework problem. You do not need to prove it again.

Proof

Any schedule with an inversion can be modified to be more like our algorithm's output without making it worse.

Algorithmic Pizza 2

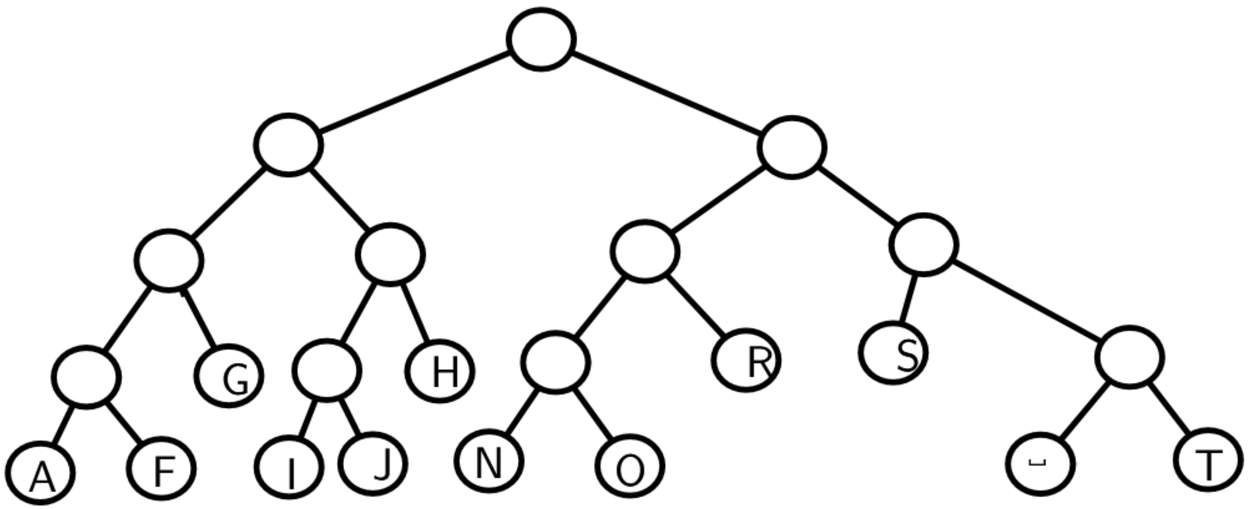
Problems with some other encodings...

- a = 0, b = 1, c = 00, d = 01, e = 10, etc
- a = 00000, b = 00001, c = 00010, ..., z = 11001
- a = 00000, b = 00001, ..., v = 10101, w = 1100, x = 1101, y = 1110, z = 1111

Binary Tree Based Codes

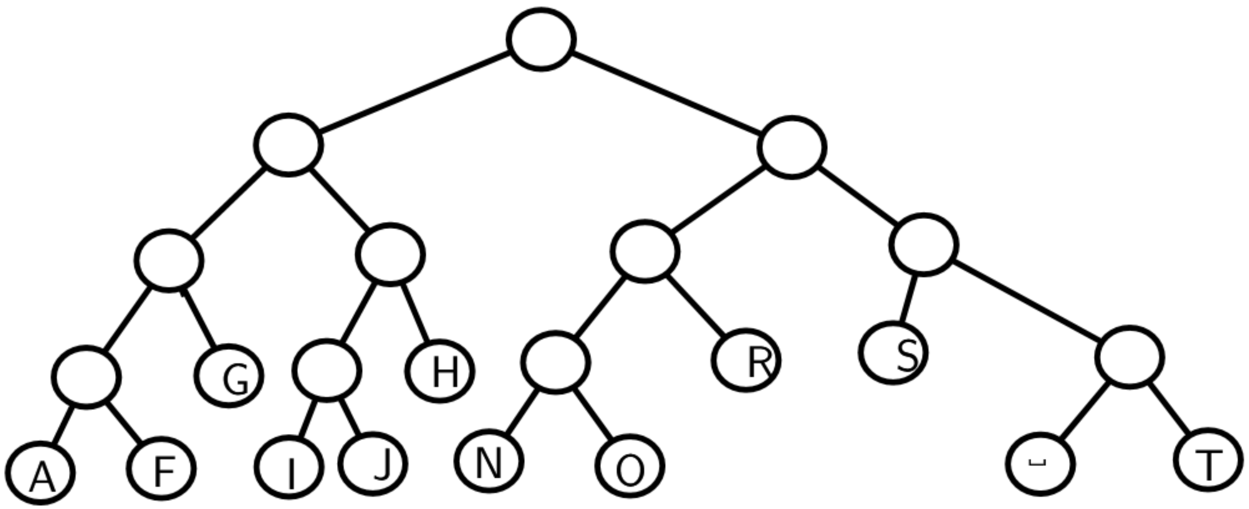
How can we use a binary tree to represent an encoding?

Binary Tree Based Codes



$$0001010000101111111110100110000 = ???$$

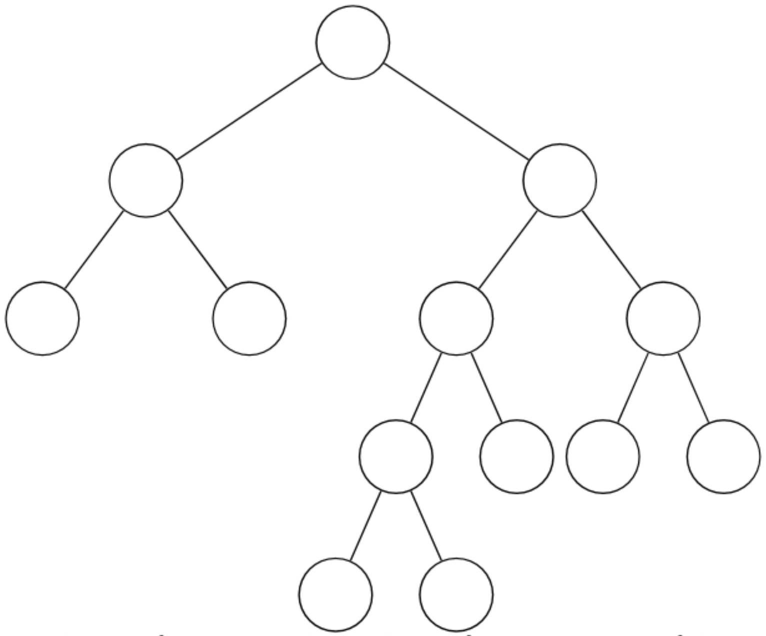
Binary Tree Based Codes



TROJANS = ???

Binary Tree Based Codes

letter x	frequency f_x
A	21%
B	18%
C	6%
D	5%
E	12%
F	23%
G	15%



Where should the letters go in order to minimize the average bit length of a compressed message?

Optimal tree for these letters?

letter x	frequency f_x
A	21%
B	18%
C	6%
D	5%
E	12%
F	23%
G	15%