

GPS 7: Mon Oct 19

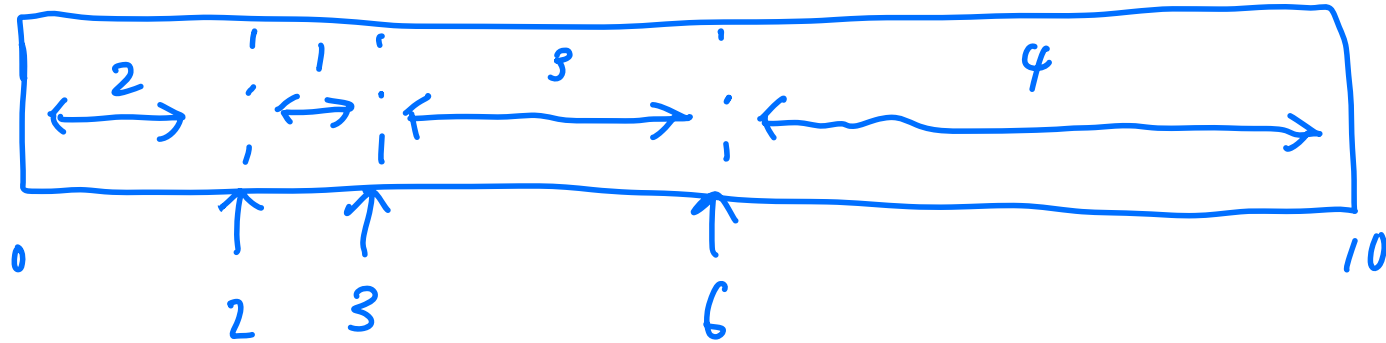
HW 7: Tue Oct 20

Drop Deadline: Fri Oct 16

Extra Office Hour: Today Oct 15 2-3pm in 2336 SC

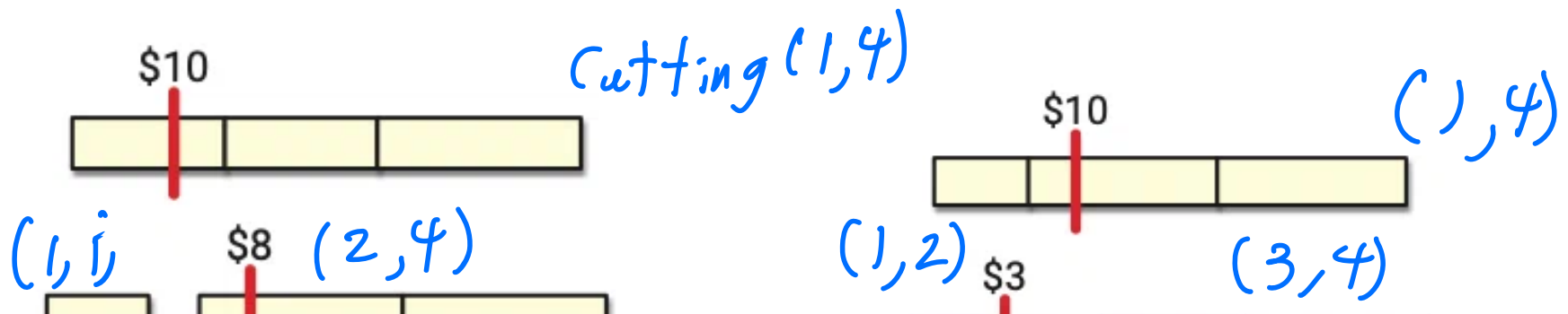
Start a single plank of wood.

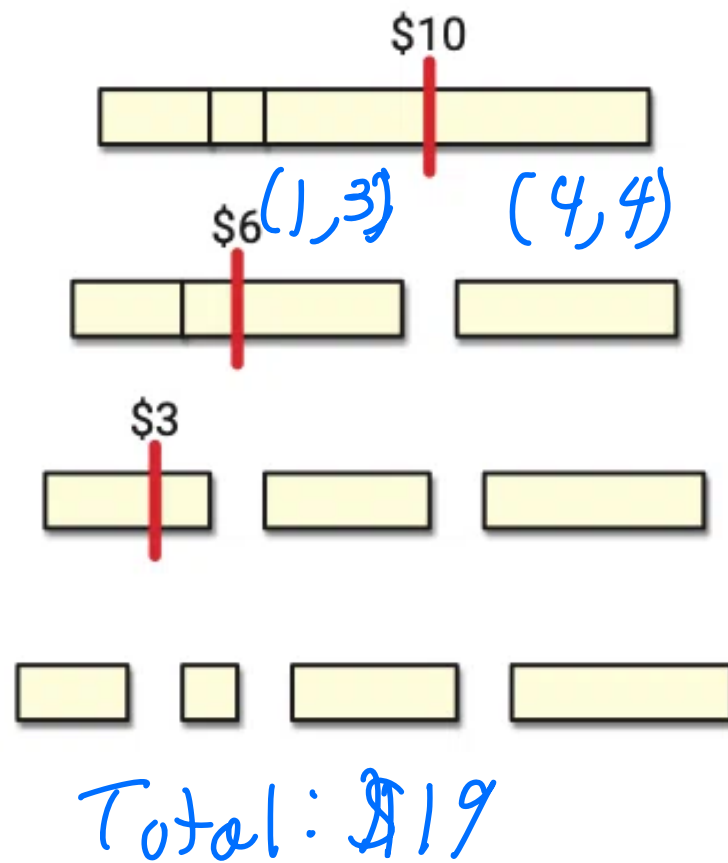
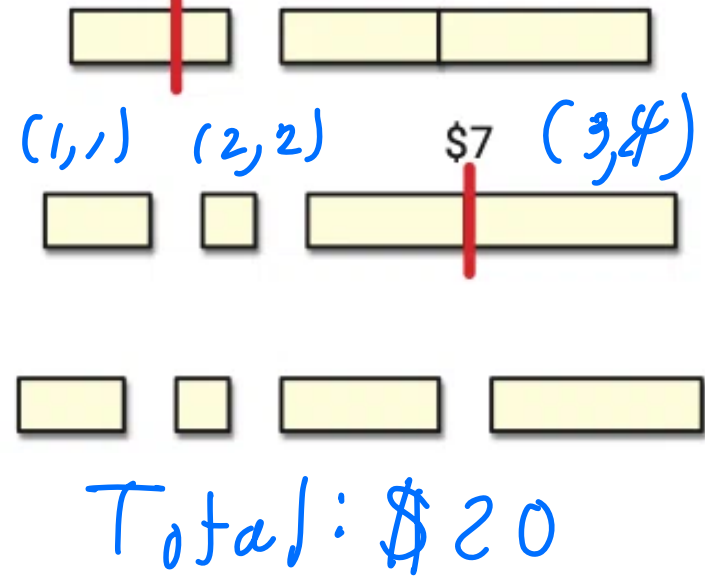
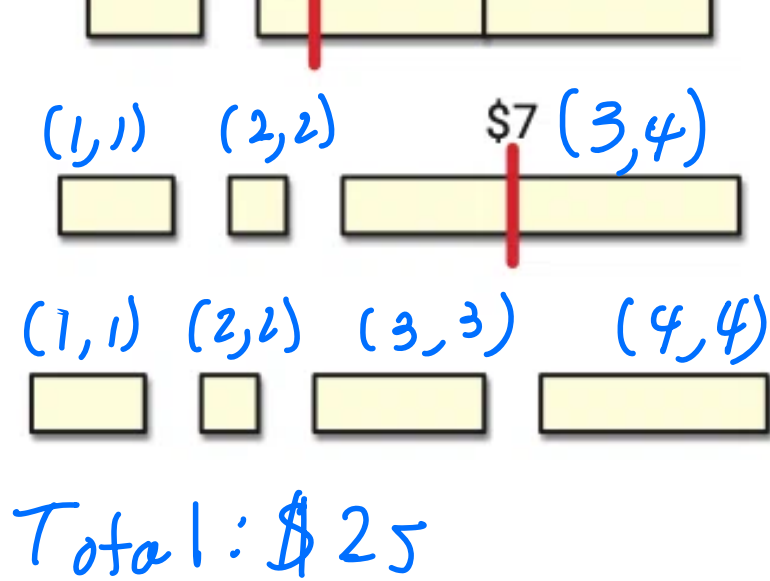
Marked at $n-1$ locations



Need to cut at all marked locations.

Cut one at a time, Charge equal to length of plank being cut.





Given $Len[1..n]$: the lengths of the final pieces in left to right order.

Above, $n = 4$ & $Len = [2, 1, 3, 4]$.

Want min possible cost for cutting the plank.

Decision: Where to make first cut?

Then recursively find best cuts for both halves.

Like building a binary tree. We have a root cut & recursive trees.

Each subproblem has a first & last
final plank piece.

Each given by a subarray $Len[i..k]$.

$Cutting(i, k)$: min cost to cut up a
plank given by lengths in $Len[i..k]$
(Assume $i \leq k$.)

Want to compute $Cutting(1, n)$.

$$Cutting(i, k) = \begin{cases} 0 & \text{if } i = k \\ \underbrace{\sum_{j=i}^k Len[j]} + \min_{i < r \leq k} \left(\underbrace{Cutting(i, r-1)}_{\text{o.w.}} + Cutting(r, k) \right) \end{cases}$$

cost of first cut for all pieces i through k start index of right subplank

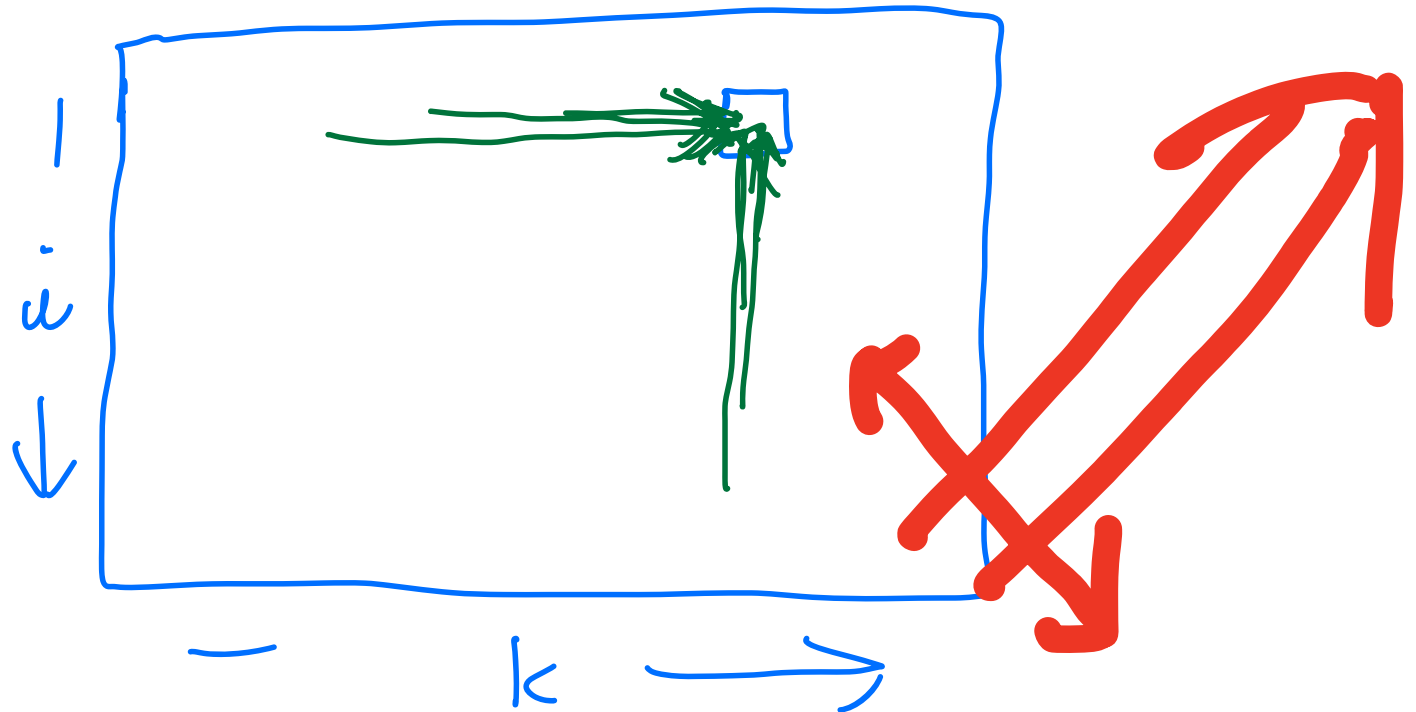
Subproblems: $1 \leq i \leq k \leq n$

Memoization: 2D array $Cutting[1..n, 1..n]$.

Dependencies:

Eval order:

Can do diagonal by

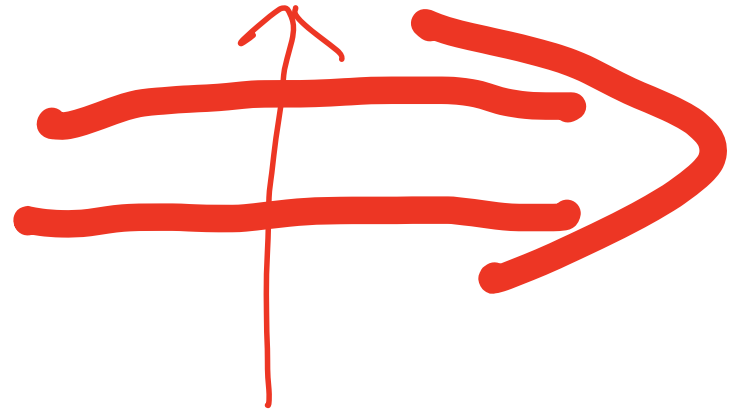


diagonal.

Also,



-or-



```
for i ← n down to 1  
  for k ← i to n
```

```
for k ← 1 to n
```

```
  for i ← k down  
    to 1
```

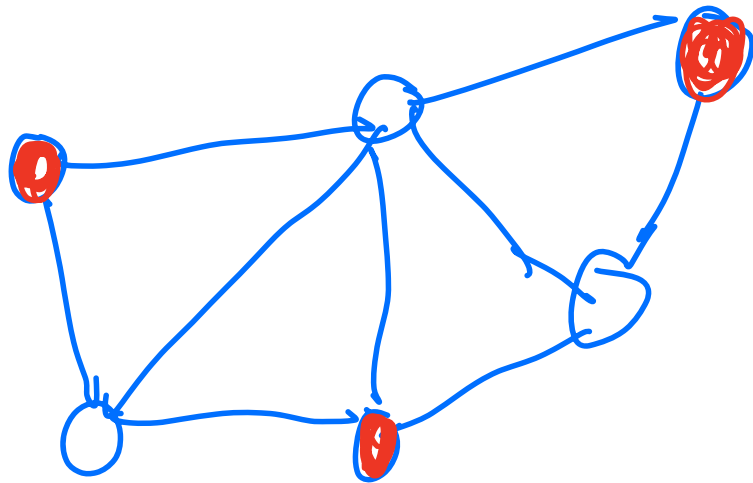
Run time: $O(n) \cdot O(n^2) = O(n^3)$

↑ ↑
time per # subproblems

See book/notes for code.

Given a graph $G = (V, E)$,
vertex edges

an independent set $S \subseteq V$ is a
subset of vertices s.t. no edge
has both endpoints in S .



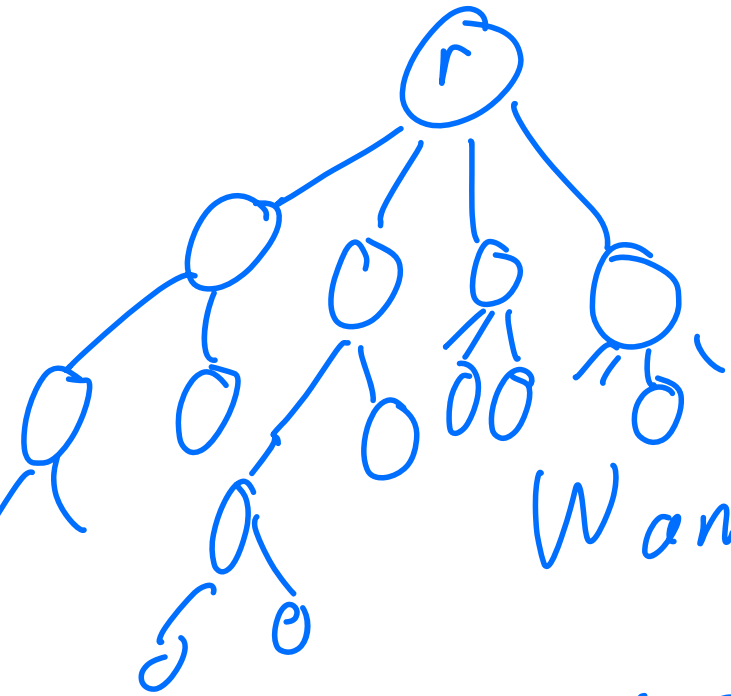
S Want a max
size independent
set.

NP-hard!

no fast
alg!
(probably)

So stick to a rooted tree T .

$n := \# \text{ nodes}$



Let r be the root of T .

Want to find max ind set S .

- If $r \notin S$. We can recursively find best ind. set for all child subtrees.
- If $r \in S$. Cannot include any child of r . Can recursively find best ind. set for grandchildren.

Subtrees + combine.

$MIS(v)$: max size of any ind. set
at the subtree rooted at v .

Want $MIS(r)$.

Let $w \downarrow v$ mean " w is a child of v ".

$$MIS(v) = \max \left\{ \sum_{w \downarrow v} MIS(w), 1 + \sum_{w \downarrow v} \sum_{x \downarrow w} MIS(x) \right\}$$

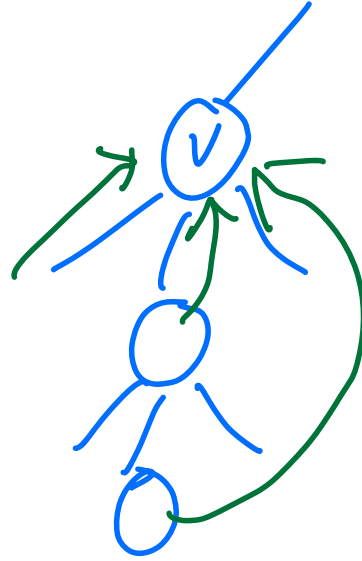
$\uparrow$
 $v \notin S?$

$\uparrow$
 $v \in S?$

Subproblems: one per node v

Memoization: Define $v, MIS := MIS(v)$
for all nodes v .

Dependencies:



Eval order:
post-order

Run time: Each node is a child $\leq$ once
a grandchild $\leq$ once
 $O(n)$ time total

Second strategy.

$MIS_{yes}(v)$: max ind. set size for
 v 's subtree. Have to include v .

$MIS_{no}(v)$: same but do not include v .

$$MIS_{yes}(v) = 1 + \sum_{w \downarrow v} MIS_{no}(w)$$

$$MIS_{no}(v) = \sum_{w \downarrow v} \max \{ MIS_{yes}(w), MIS_{no}(w) \}$$

Still $O(n)$ time.