CSCI 3104: Algorithms

Lecture 21: Dynamic Programming – Wrap Up of Topics

Rachel Cox

Department of Computer Science

What will we learn today?

- ❑ Rod Cutting
- ❑ Topological Sort

Intro to Algorithms, CLRS:
Sections 24.1

↑ schedule

Dijkstra's Algorithm

Algorithm SSSP-Dijkstra (G, s)

1. Q = ∅ //** Q is a priority queue
2. d[s] = 0 and insert s into Q
3. for each v ≠ s
4. d[v] = ∞
5. insert v into Q with key d[v]
6. while Q ≠ ∅
7. u = extract-min(Q)
8. for each neighbor of u
9. d[v] = min{d[v], d[u] + wt(u,v)}
10. endfor
11. endwhile

Recall
- only for positive edges.
- Dijkstra's does not ensure that edges are examined in any particular order.
- It does ensure that when node is removed from Q, we have shortest path to that node

Runtime:

- V - node set
- E - edge set

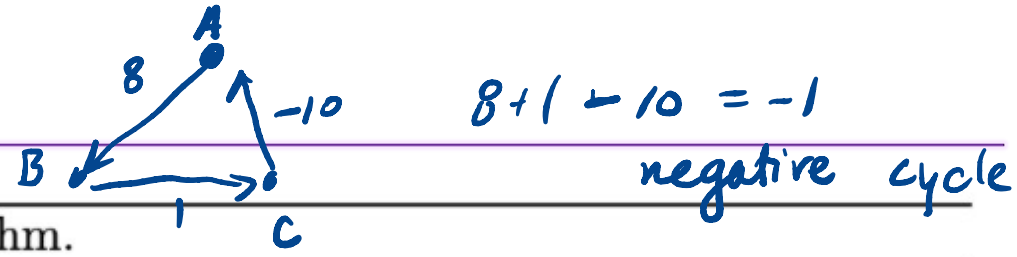
If use a binary heap to do "pop" from Q, ⇒ $O(\log V)$ time

while loop: $O(V \log V)$

$O(E \log V)$

⇒ $O((|V| + |E|) \log V)$

Bellman–Ford Algorithm



Algorithm 7 The Real Bellman–Ford algorithm.

procedure BELLMANFORD($G = (V, E, w), r$)
  let P be an array of length n containing the element NULL
  let D be an array of length n containing the element ∞
  $D[r] = 0$

  for i in $1..|V| - 1$ **:** #*i.e. repeat* $|V| - 1$ *times*
    for every edge (u, v) in E **:**
      if $D[v] > D[u] + w(u, v)$ **:** #*if* (u, v) *is tense*
        $P[v] = u$ #*relax* (u, v) *by updating* P
        $D[v] = D[u] + w(u, v)$ #*and by updating* D
  for every edge (u, v) in E **:**
    if $D[v] > D[u] + w(u, v)$ **:** #*if* (u, v) *is still tense*
      throw "Negative cycle!"

- Use when we have negative edges
- Can "handle" negative negative cycles
- Slower than Dijkstra

Runtime: $O(|V| \cdot |E|)$

- More expensive than Dijkstra's but it is more general

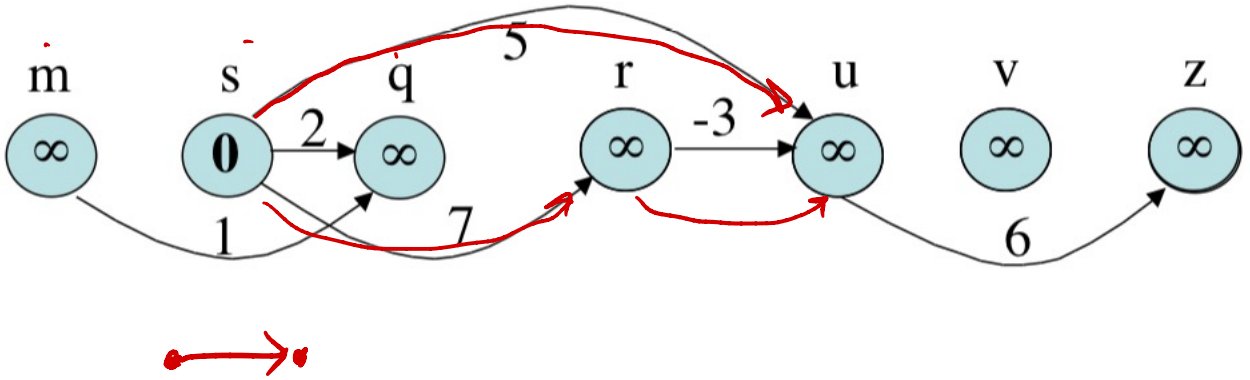
Topological Sort

Note: • Dijkstra's can handle a positive cycle
• B-F can handle a negative cycle.

If the graph is a DAG (Directed Acyclic Graph), we can use a topological sort on the vertices and compute the shortest path from a single source in $O(|V| + |E|)$ time.

linear!!

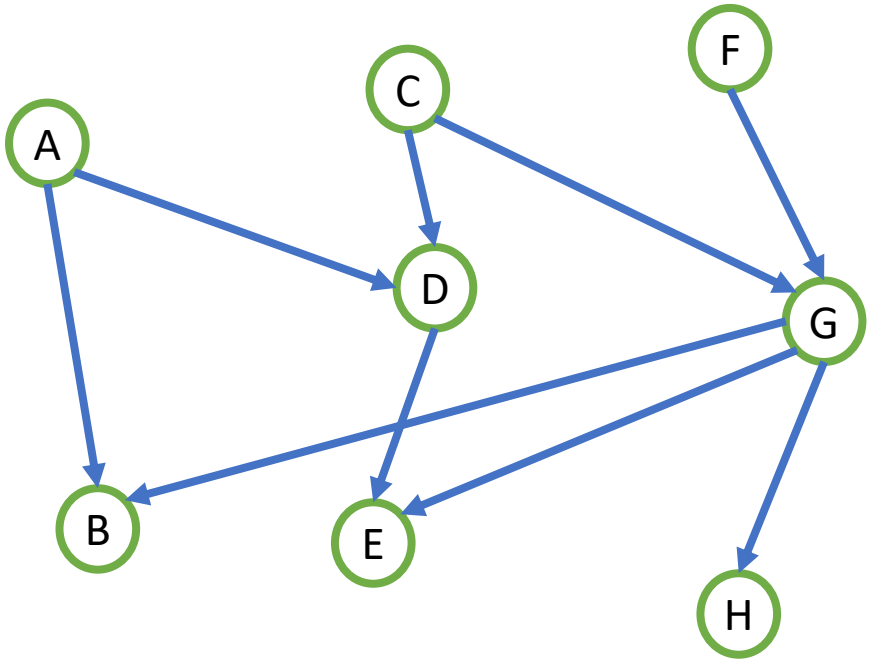
In a topologically sorted list of vertices, all edges will go from left to right. Once all outgoing edges at a node u have been relaxed, u will never be revisited. Since we process the modes in topologically-sorted order, the nodes at one hop from s will be finished before those at 2 hops, etc. By the path relaxation property, all shortest paths will be found.



Source

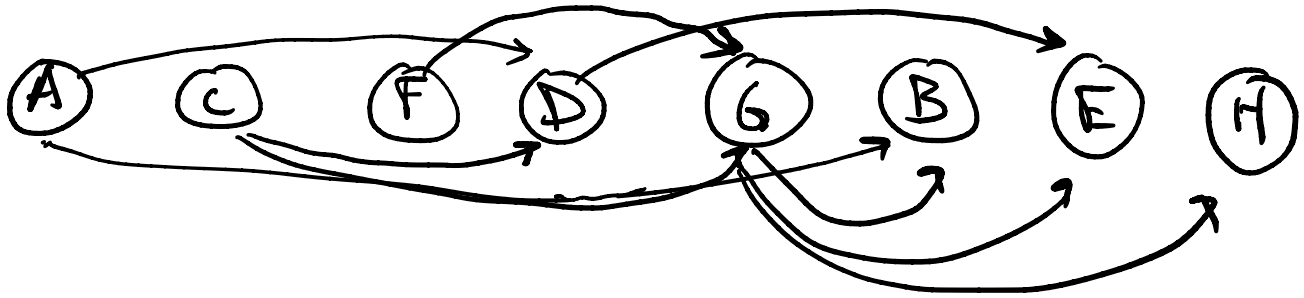
Topological Sort

Topological sorting for a DAG is a linear ordering oof vertices such that for every directed edge (u, v), vertex u comes before vertex v in the ordering.



one possible topological sort:

A, C, F, D, G, B, E, H



Another one: F, C, A, G, D, H, E, B

- many sorts!

Topological Sort

- Let V be the list of vertices in a graph in topological order.
- Initialize distance to source as 0, initialize distance from source to all other vertices as ∞
- Loop over vertex list V, set current vertex = u
  - For each vertex v, with edge (u, v)
  - if $d(u) + c(u, v) < d(v)$:
    - relax the edge!

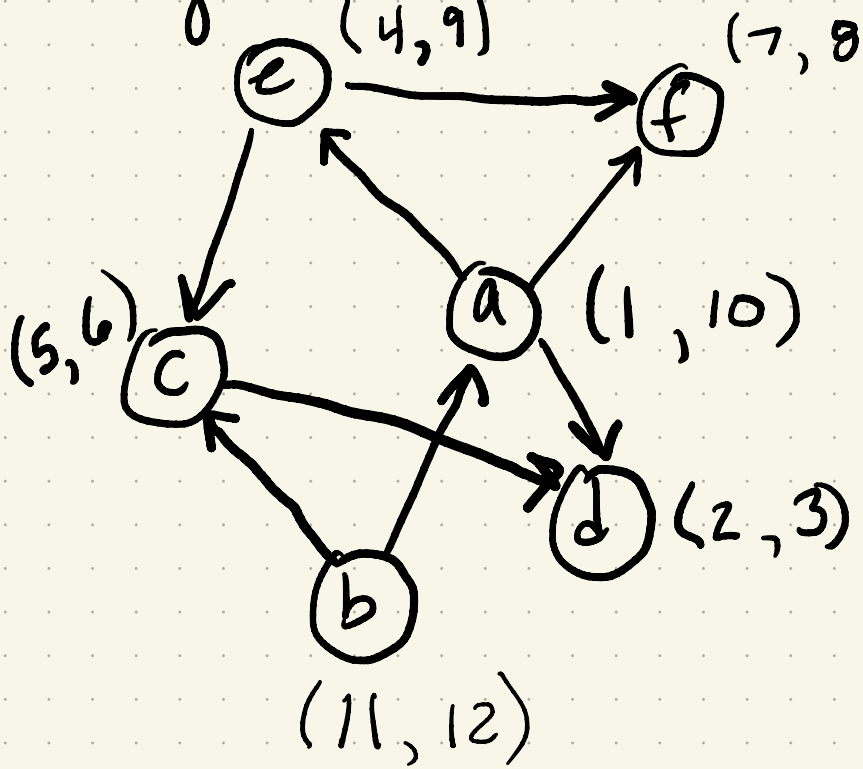
Single-Source Shortest Path problem

On a graph of n vertices and m edges, what is the running time?

$O(n+m)$ linear!

Algorithm Idea for how to topologically sort.

- can only be done on DAGs

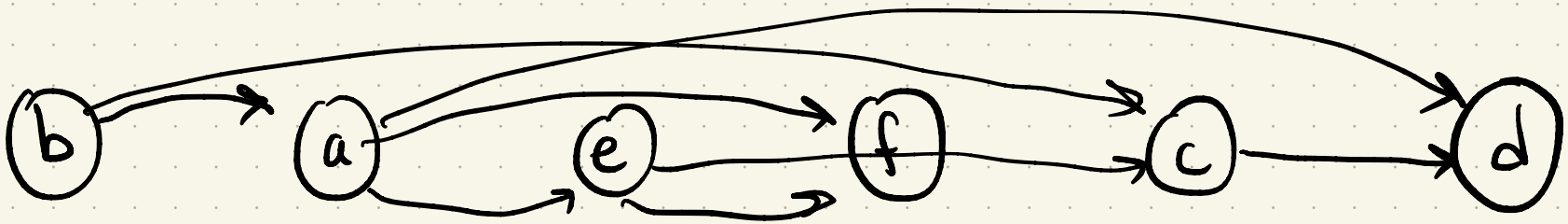


- pick a node to start from.
- (start time, finish time)

Start from (a)

List the nodes by their end time in decreasing order

b, a, e, f, c, d

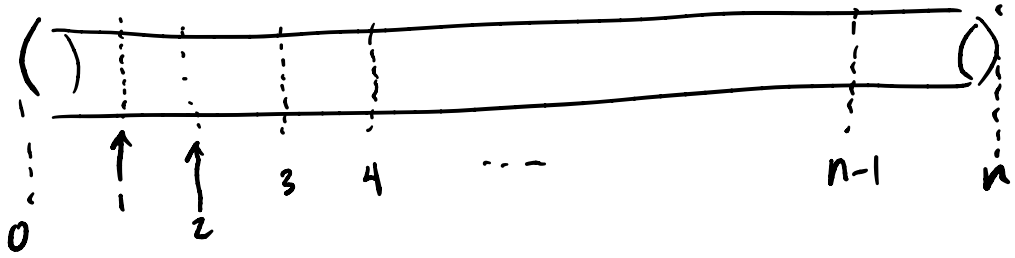


Rod Cutting

Rod Cutting Problem:

- Idea: We have a rod of length n which will be cut into integer-length pieces.
- The pieces will be sold. In general, longer pieces can be sold for more than shorter pieces.
- Let $p_0, p_1, \ldots, p_n$ be a list of prices where $0 = p_0 < p_1 < \ldots < p_n$

How many ways are there to cut up a rod of length n?



How many places can you make cuts in an n-length rod?

n-1 places

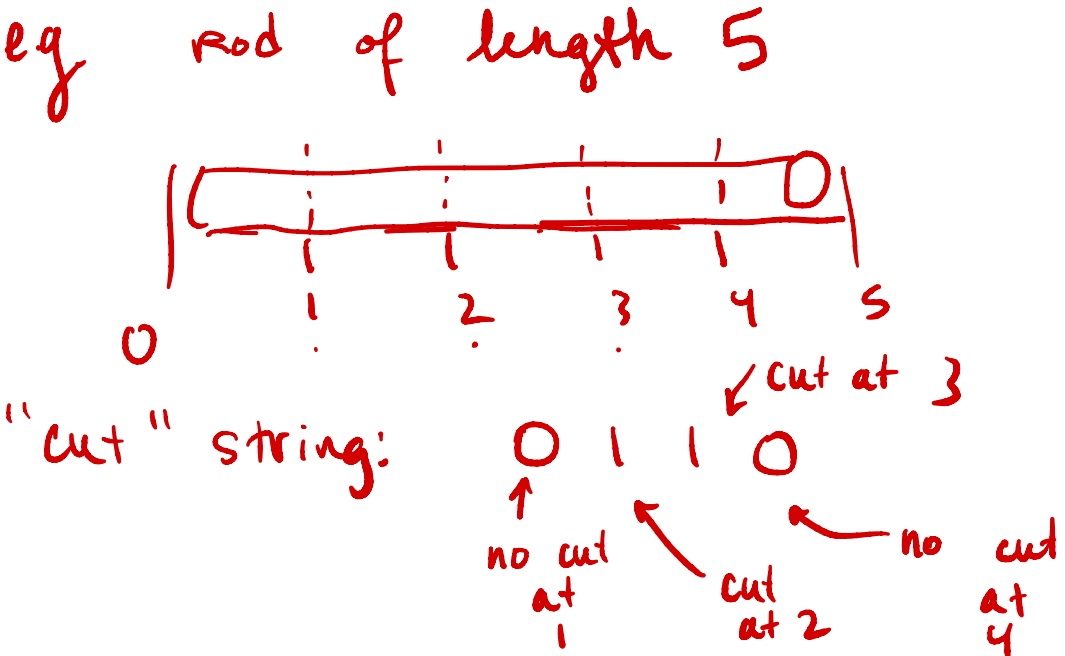
- Imagine we have a binary string that stores info on whether we made a cut at a particular integer value or not.

Rod Cutting

Rod Cutting Problem:

- Idea: We have a rod of length n which will be cut into integer-length pieces.
- The pieces will be sold. In general, longer pieces can be sold for more than shorter pieces.
- Let $p_0, p_1, \ldots, p_n$ be a list of prices where $0 = p_0 < p_1 < \ \ldots < p_n$

How many ways are there to cut up a rod of length n?



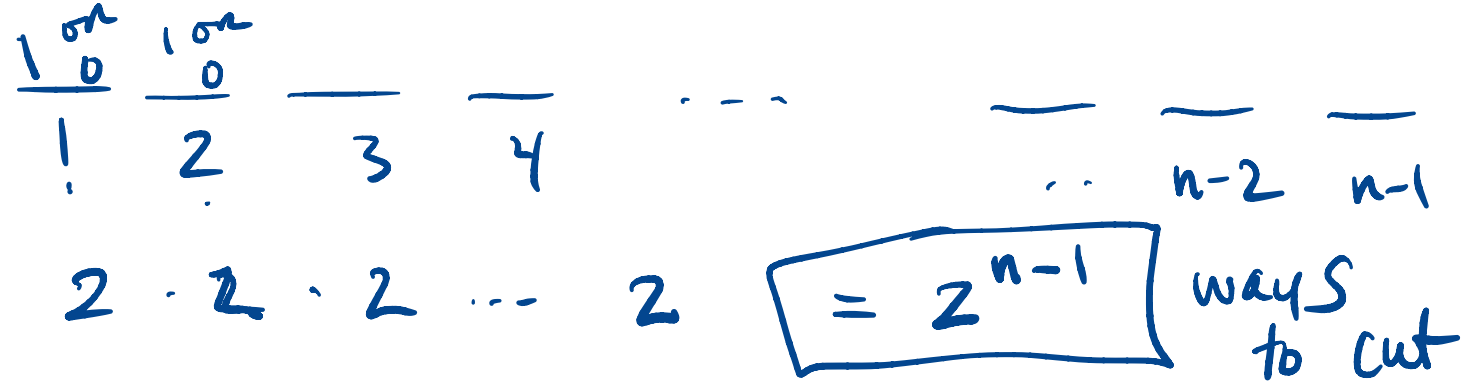
Rod Cutting

Rod Cutting Problem:

- Idea: We have a rod of length n which will be cut into integer-length pieces.
- The pieces will be sold. In general, longer pieces can be sold for more than shorter pieces.
- Let $p_0, p_1, \ldots, p_n$ be a list of prices where $0 = p_0 < p_1 < \ \ldots < p_n$

How many ways are there to cut up a rod of length n?

So for an n-length rod, we make a decision at each of the n-1 possible cut locations: cut? or no cut?



Rod Cutting

Rod Cutting Problem:

- Idea: We have a rod of length n which will be cut into integer-length pieces.
- The pieces will be sold. In general, longer pieces can be sold for more than shorter pieces.
- Let $p_0, p_1, \ldots, p_n$ be a list of prices where $0 = p_0 < p_1 < \ \ldots < p_n$

How many ways are there to cut up a rod of length n?

Problem: maximize the value of the rod by cutting it into pieces to sell.

main Recursive piece: $r_n = \max_{1 \le i \le n} (p_i + r_{n-i})$

p_i ← piece of Rod

r_{n-i} ← recursive call

Recursive algorithm

```
def rod_cut(P, n)
    if n == 0
        return 0
    q = 0
    for i = 1 to n:
        q = maximum(q, P[i] + rod_cut(P, n-i))
    return q
```

P ← profit function

n ← length of rod

* top down approach

* Bottom up approach

```
def rod_cut2(P, n)
    r                          - revenue array   n+1 length array
    r[1] = 0

    for j in range(2 : n+1):
        initialize q = -∞
        for i in range(1, j):
            q = max(q, P[i] + r[j-i])
        r[j] = q
    return r
```

- outer loop iteratively cut the rod
- inner loop is maximizing how to cut and sell a rod of length j

Rod Cutting

Suppose you have started a business selling fresh rocky mountain water to out-of-state breweries. You have 100 gallons of water, and you sell your water in 10 gallon buckets. Notice that this means you have 10 buckets to sell. Your buyers are all willing to buy any amount of water that you will sell them. Assuming you can only distribute the 100 gallons of water in the 10 gallon buckets, reference the below table for the market value of various rocky mountain water amounts. b buckets of water can be sold for d_b dollars. Notice that if you sell 2 buckets to a buyer at once (for 20), you only have 8 more buckets to sell.

P:

buckets b	1	2	3	4	5	6	7	8	9	10
dollars d_b	5	20	40	40	45	50	55	80	100	120

we're reducing "buckets of water problem" to Rod cutting

(a) How many different ways you can distribute your 10 buckets of water (Assuming you must distribute all 10 buckets).

e.g. sell 1 bucket to one buyer
sell 9 buckets to another
etc...

$2^{10-1} = 2^9$ = 512 ways

Rod Cutting

exercise for coding!

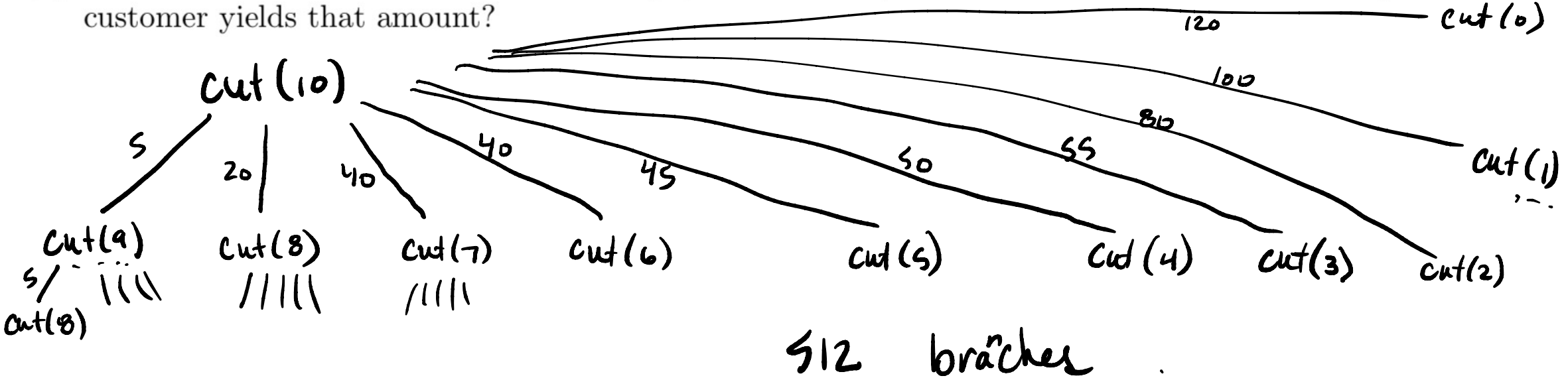
Suppose you have started a business selling fresh rocky mountain water to out-of-state breweries. You have 100 gallons of water, and you sell your water in 10 gallon buckets. Notice that this means you have 10 buckets to sell. Your buyers are all willing to buy any amount of water that you will sell them. Assuming you can only distribute the 100 gallons of water in the 10 gallon buckets, reference the below table for the market value of various rocky mountain water amounts. b buckets of water can be sold for d_b dollars. Notice that if you sell 2 buckets to a buyer at once (for 20), you only have 8 more buckets to sell.

buckets b	1	2	3	4	5	6	7	8	9	10
dollars d_b	5	20	40	40	45	50	55	80	100	120

3,3,3,1 bucket splits

3 * $40 + 1 * $5 = $125

(b) What is the maximum amount of money you can make, and what set of bucket amounts per customer yields that amount?



512 branches.

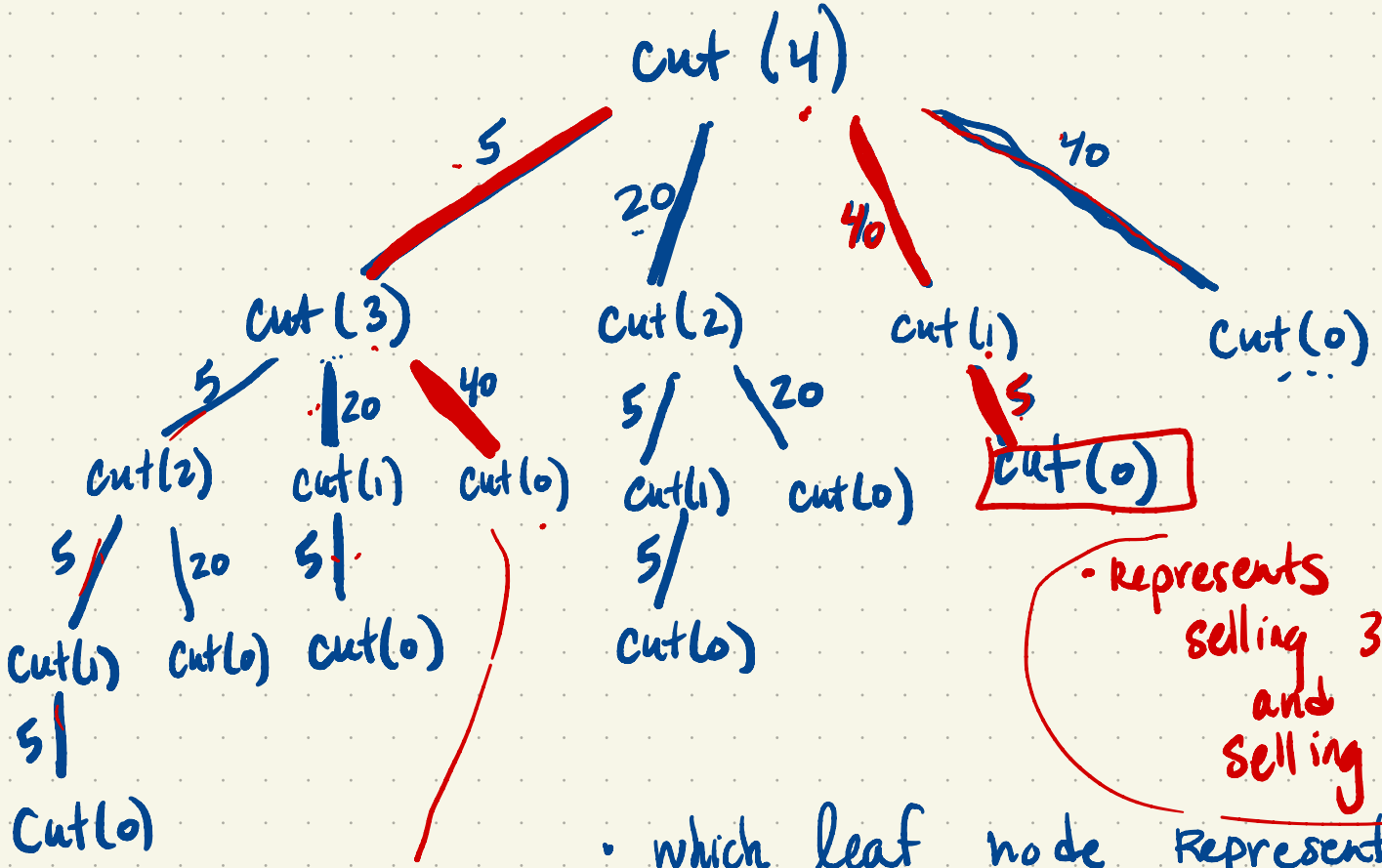
Let's make the previous question easier.
Let's consider 40 gallons to distribute

	1	2	3	4
	5	20	40	40

1 gallon $5
3 gallons + $40
$45

or

2 gallons $20
1 gallon $5
1 gallon $5
$30



Represents selling 1 bucket

Selling 3 buckets

- Represents selling 3 buckets and selling 1 bucket

- which leaf node Represents the set of optimal bucket distributions?

$45 (4,3,0) (4,1,0)

Rod Cutting

code this in python

```
def groupBuckets(table, n):
    buckets ← [1, .., n];
    // Initialize first index of buckets to 0 to handle i == j case
    buckets[0] ← 0;
    // For tracking the maximum dollar amount
    max_d ← −∞;
    for i in 1..n do
        for j in 1..i do
            // Compare dollars from j buckets + optimal remainder
            max_d ← max(max_d, table[j] + buckets[i − j]);
        end
        buckets[j] ← max_d;
    end
    return buckets[n]
```

Algorithm 1: DynamicSolution