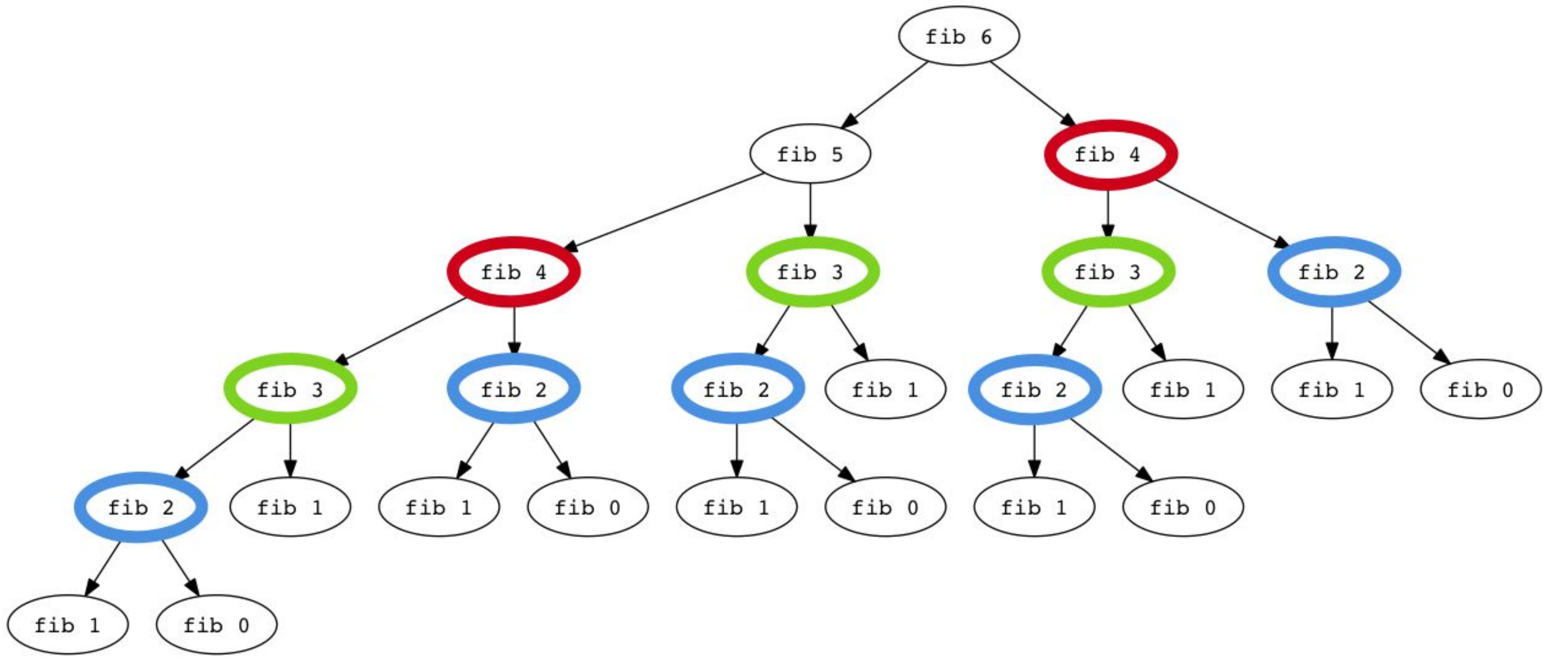
CS180 Discussion

Week 5

Lecture Recap

- Dynamic programming
- Knapsack
- Longest common subsequence
- Divide and conquer
- Merge-sort

Dynamic Programming



Knapsack

$$OPT(i,w)=\begin{cases}0 & \text{if } i=0\\ OPT(i-1,w) & \text{if } w_i > w\\ \max\{\,OPT(i-1,w),\ \ v_i + OPT(i-1,w-w_i)\,\} & \text{otherwise}\end{cases}$$

(Val / Wt)	0	1	2	3	4	5	6
(1/1)	0	1	1	1	1	1	1
(4/3)	0	1	1	4	5	5	5
(5/4)	0	1	1	4	5	6	6
(7/5)	0						
(8/6)	0						

Knapsack

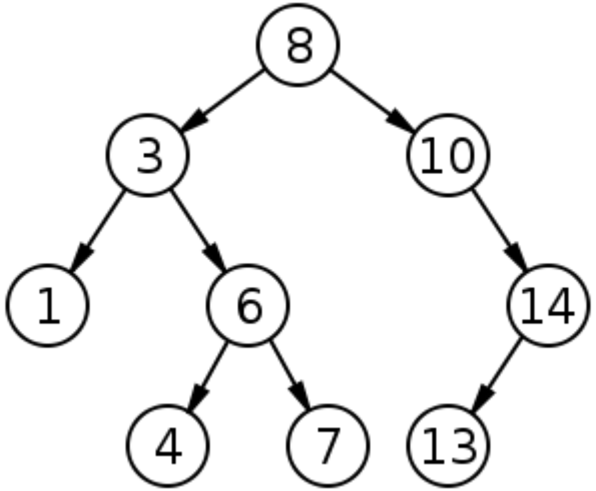
$$OPT(i,w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,w) & \text{if } w_i > w \\ \max\{\, OPT(i-1,w),\ \ v_i + OPT(i-1, w-w_i) \,\} & \text{otherwise} \end{cases}$$

(Val / Wt)	0	1	2	3	4	5	6
(1/1)	0	1	1	1	1	1	1
(4/3)	0	1	1	4	5	5	5
(5/4)	0	1	1	4	5	6	6
(7/5)	0	1	1	4	5	7	8
(8/6)	0	1	1	4	5	7	8

Longest Common Subsequence

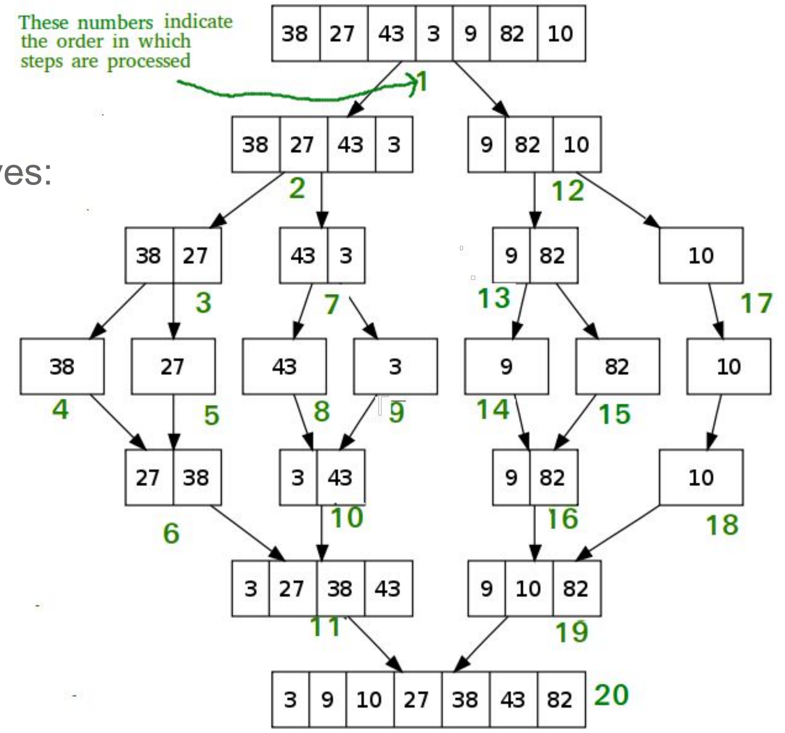
		0	1	2	3	4	5	6	7
		Ø	M	Z	J	A	W	X	U
0	Ø	**0**	0	0	0	0	0	0	0
1	X	0	0	0	0	0	0	1	1
2	M	0	**1**	1	1	1	1	1	1
3	J	0	1	1	**2**	2	2	2	2
4	Y	0	1	1	2	2	2	2	2
5	A	0	1	1	2	**3**	3	3	3
6	U	0	1	1	2	3	3	3	**4**
7	Z	0	1	2	2	3	3	3	4

Divide and Conquer



Merge Sort

```
MergeSort(arr[], l,  r):
If r > l
     1. Find the middle point to divide the array into two halves:
             middle m = (l+r)/2
     2. Call mergeSort for first half:
             Call mergeSort(arr, l, m)
     3. Call mergeSort for second half:
             Call mergeSort(arr, m+1, r)
     4. Merge the two halves sorted in step 2 and 3:
             Call merge(arr, l, m, r)
```



Q

It takes n-1 comparisons to find the minimum number in a given list of integers L = (x1, x2, x3, ...). Similarly, it takes n-1 comparisons to find the maximum. Therefore, it is trivial to design an algorithm that finds both the minimum and the maximum with about 2n-2 comparisons. Design an algorithm to find both the minimum and the maximum in a list using about 3n/2 comparisons.

Solution

- First, divide items into n/2 pairs
- Compare items within each pair
- Then,
    - Assignment = (n/2)
    - max= (n/2)-1 comparisons with larger items
    - min= (n/2)-1 comparisons with smaller items
- Total (3n/2)-2 comparisons

terview Questions!!!

Interview Questions!!!

Interview Questions!!!

Interview Questions!!!

Credit card collection

take two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of n cards, is there a set of more than n/2 of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only O(n log n) invocations of the equivalence tester.