Final Review

Data Structures and Algorithms

Announcements

Final Review Due Tonight – Don't stress too much about it!

Final on Friday – 1 page of notes is allowed

Course evaluations due tomorrow night. (see e-mail)

Please fill out the survey on Kendra Yourtee's talk, and sign her thank-you card!

Dynamic Programming

Subsequence palindrome:

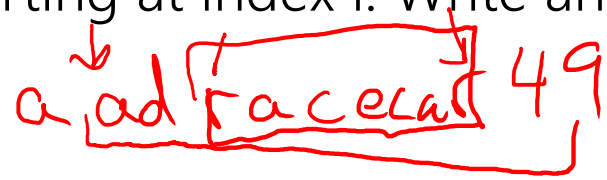
Given a string S, find the longest **subsequence** that is a palindrome.

RACECAR

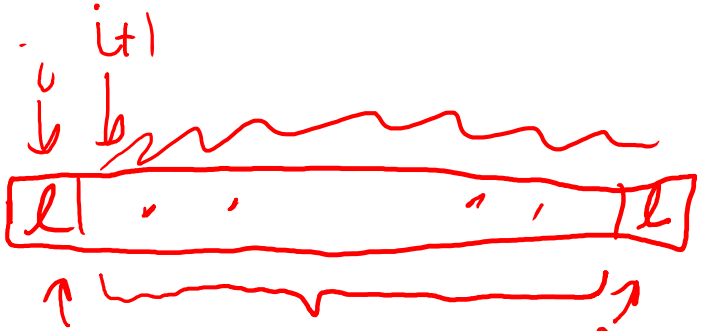
Unlike your homework, the letters don't need to be consecutive!

Dynamic Programming

Let *OPT(i, n)* denote the length of the longest palindrome *subsequence* in the substring of length n starting at index i. Write an expression for the recursive case of *OPT(i, n)*.



OPT(1, 10)



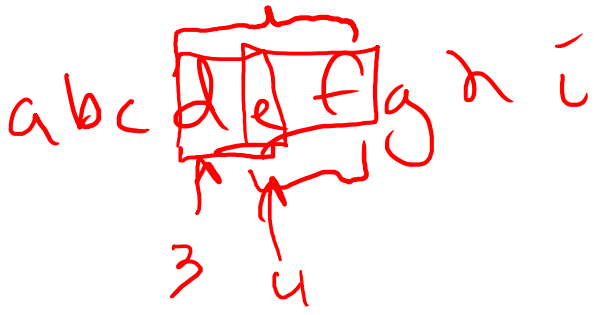
$OPT(i,n) = 2 + OPT(i+1, n-2)$ if $S[i] == S[i+(n-1)]$

both first & last characters are in the palindrome
first char. not in palindrome
last char. not in palindrome

Dynamic Programming

Let *OPT(i, n)* denote the length of the longest palindrome *subsequence* in the substring of length n starting at index i. Write an expression for the recursive case of *OPT(i, n)*.

$$OPT(i,n) = \begin{cases} 2 + OPT(i+1, n-2), & S[i] = S[i+n-1] \\ \max\{OPT(i,n-1), OPT(i+1,n-1)\}, & otherwise \end{cases}$$



OPT(3, 3)

Dynamic Programming

Next we need a base case for our *OPT* recurrence. Write an expression for the base case(s) of this recurrence.

Dynamic Programming

Next we need a base case for our *OPT* recurrence. Write an expression for the base case(s) of this recurrence.

$$OPT(i, 0) = 0$$

$$OPT(i, 1) = 1$$

Dynamic Programming

Now that we have a complete recurrence, we need to figure out which order to solve the subproblems in. Which subproblems does the recursive case *OPT(i, n)* require to be calculated before it can be solved?

$$OPT(i+1, n-2), OPT(i, n-1), OPT(i+1, n-1)$$

Dynamic Programming

Given these dependencies, what order should we loop over the subproblem in?

Dynamic Programming

Given these dependencies, what order should we loop over the subproblem in?

Since we depend on OPT(i+1, **n-2**), OPT(i, **n-1**), and OPT(i+1,**n-1**) it is sufficient to have solved all subproblems with smaller *n* first (i.e. the subproblem for strings of shorter length). Therefore we can loop over the subproblems in order of increasing length (the order of i does not matter).

Dynamic Programming

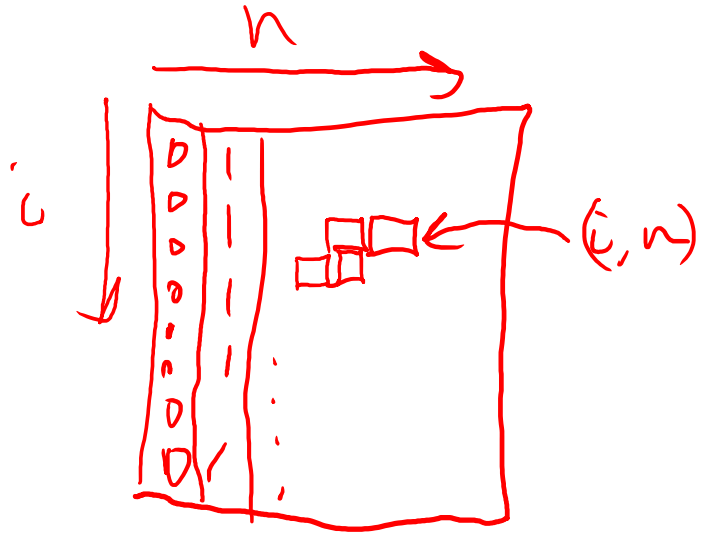
We have all of the pieces required to put together a dynamic program now. Write psuedocode for the dynamic program that computes the length of the longest palindromic *subsequence* of *S*.

Dynamic Programming

We have all of the pieces required to put together a dynamic program now. Write psuedocode for the dynamic program that computes the length of the longest palindromic *subsequence* of *S*.

```
Initialize OPT[S.length][S.length+1] with 0s for all OPT[i][0] and 1s for all OPT[i][1]
for n from 2 to S.length:
    for i from 0 to S.length – n:
        if (S[i] == S[i + n – 1]):
            OPT[i][n] = 2 + OPT[i+1][n-2]
        OPT[i][n] = max( OPT[i][n], OPT[i+1][n-1], OPT[i, n-1] )
return OPT[0][S.length]
```

addr



Differences from Your HW Review

In the HW, there is an extra constraint that the palindrome is consecutive.

You'll need an extra condition to account for this.

Sorting

A store stocks its cereal boxes in alphabetical order along the shelf from left to right. One day, a customer picks up a few boxes, and puts them back in the wrong positions. Which sorting algorithm would be best for the store to use to put the cereal isle back in order?

- only one sorter — not a parallel task
  - not too many people
- mostly sorted input
- in place - probably no other shelves

Sorting

A store stocks its cereal boxes in alphabetical order along the shelf from left to right. One day, a customer picks up a few boxes, and puts them back in the wrong positions. Which sorting algorithm would be best for the store to use to put the cereal isle back in order?

Assumptions:

a) The store doesn't have an empty shelf elsewhere to sort into.

- We want an in-place sort

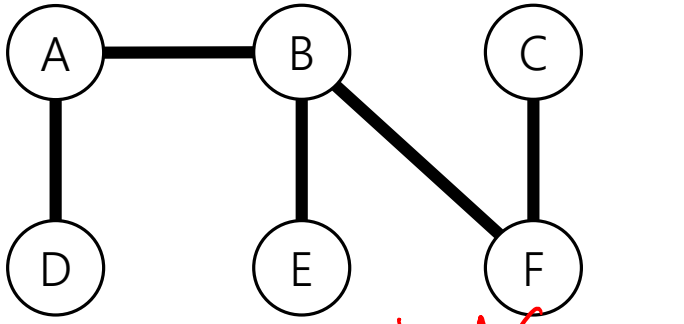
b) Only a few boxes were misplaced. The shelf is mostly sorted – only a few out-of-place.

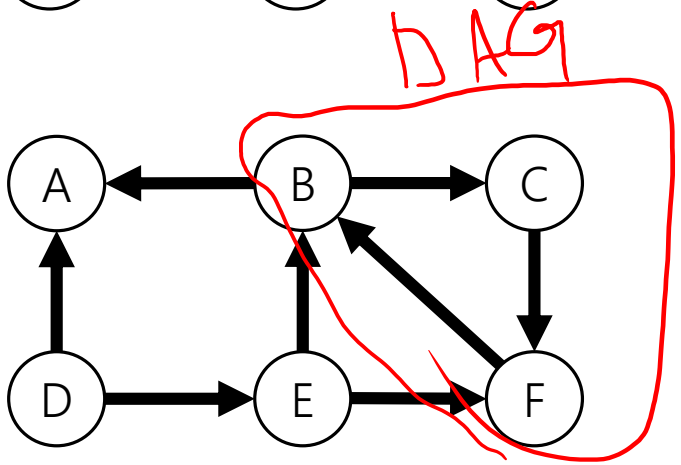
- We want a sort that has a fast best-case time when the list is mostly sorted.

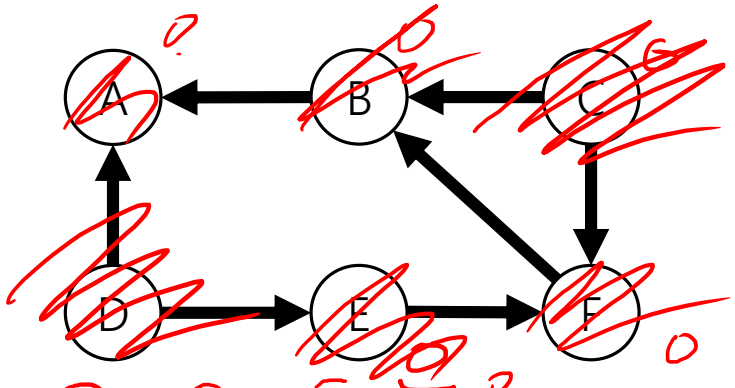
Insertion sort has both of these properties!

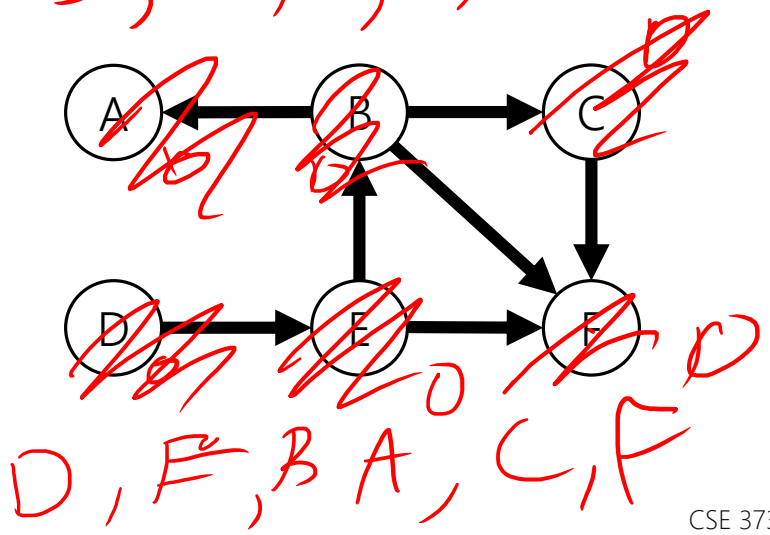
Topological Sort

For each graph, give the topological sorting, or if there is none, why?

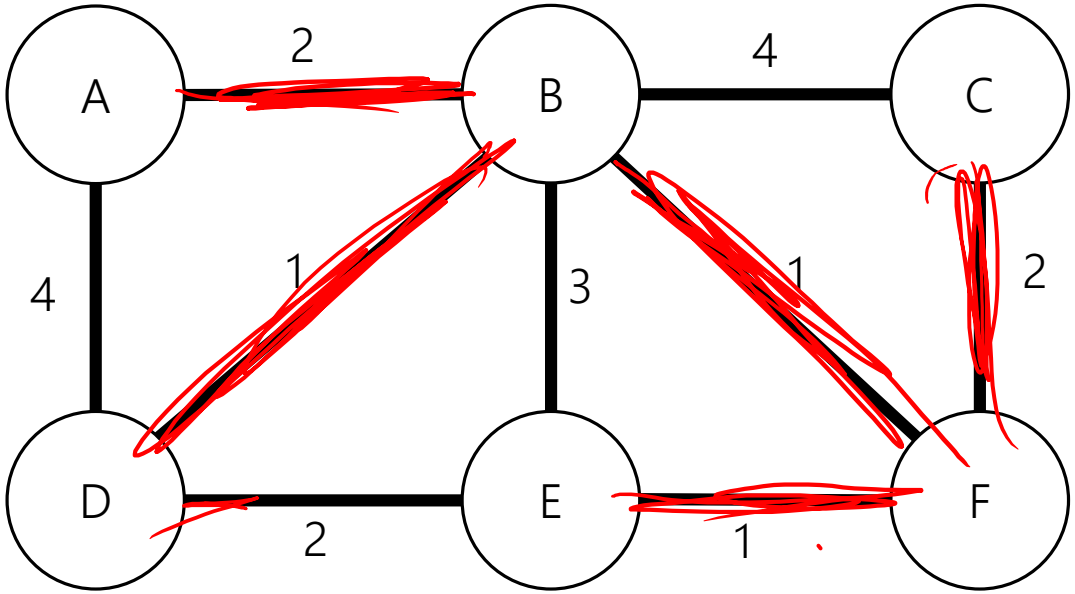








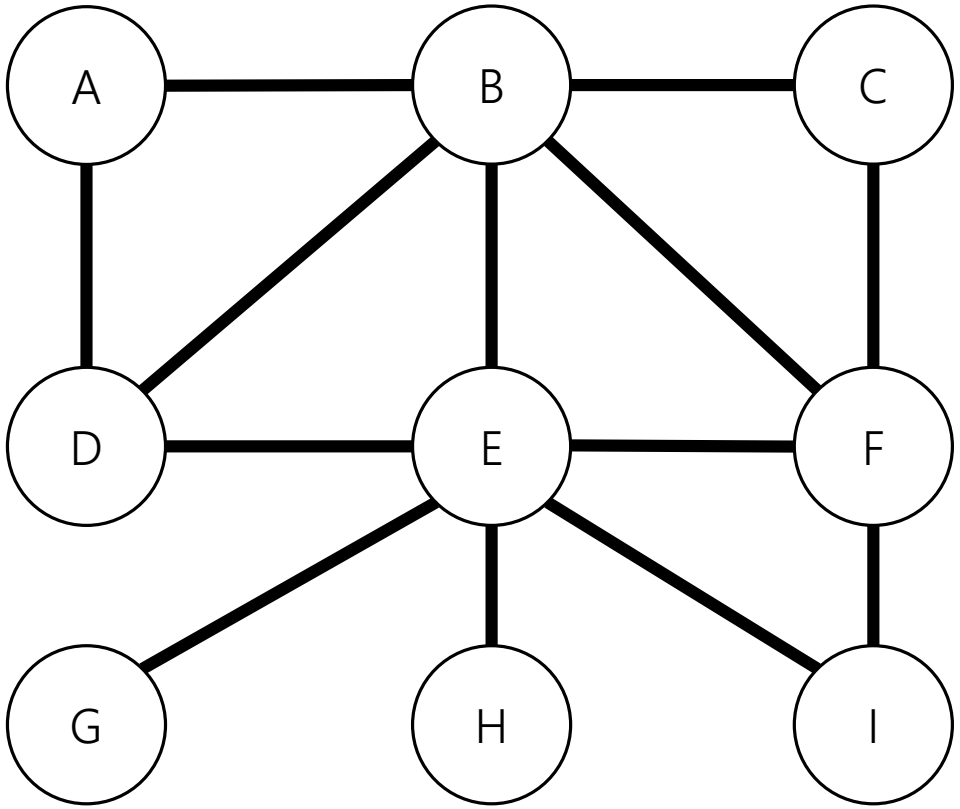
MST Prim's Algorithm



Vertex	Known?	Cost (d)	Parent
A	✓	0	—
B	✓	2	A
C	✓	2	F
D	✓	1	B
E	✓	1	F
F	✓	1	B

Graph Traversals (DFS)

Perform a DFS from vertex G. Visit neighbors in alphabetical order. Show your work.



Graph Traversals (BFS)

Perform a BFS from vertex G. Visit neighbors in alphabetical order. Show your work.