Lecture 27

CSE 331

Nov 5, 2018

Hope y'all had fun!

Video due TODAY

note ☆ stop following **150 views**

You can submit mini project video now

You can now submit your mini project videos now. It is due in a bit over 2 weeks: by **11:59pm on Mon, Nov 5**.

The mini-project page has all the details on what is needed in the submission.

Some important points:

- Please make sure you read through the instructions/requirements carefully.
  - Till last year there used to be an intermediate report stage where I could give some preliminary feedback so that y'all could avoid some of the common mistakes in the video. Y'all do not have the luxury, so please make sure you read through the page very very carefully.
- This is a **group submission**. Please see the instructions at the end of this post.
  - Main thing: do **NOT** submit your report till your group is formed.
- **Check on your group**. We are getting close to the resign date. Unfortunately, some students will drop-- so make sure you check with your group mates to see if they'll be around.
  - If your group-mate(s) drop out, then it is OK for you to continue with a smaller group.
    - Even a group of size 1 is OK if you're fine with it. But if not AND if you give me enough notice, I can try and re-assign you to another group

Peer evaluation due Wed 11:59pm

note

stop following 108 views

Peer evalutation for mini project (pleas

Peer evaluation for mini project is now live on Autolab. ground on this.

We are doing this for the first time in CSE 331 a gle with it a bit. So **we would really appreciate it if y'all cou** u have submitted the video and can answer the question ure there are no bugs etc.

Some important remarks:

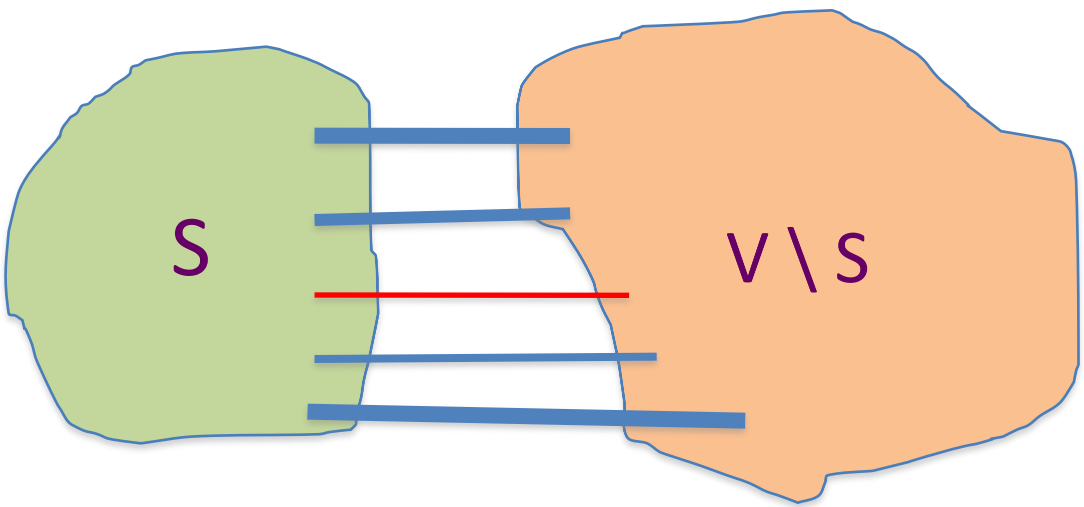
- There is some checking being done on Autolab regarding your input (specifically the UBIT IDs of your group mates) but you will not see any of those when you fill in the form, which is static.
  - **Please be sure to check the feedback** (by clicking on numbers like you usually do for Q1) to see if there are any issues.
- **If one of your group-mates have dropped, please test out the system by FRIDAY and let me know if you still have a member showing up in the feedback who should not be there.** The start of the feedback will list the UBIT IDs of your group mates.
  - I we ut but I'm prett
- You will n
  - If yo dback.
- **The scores that you see are NOT your final scores.**
  - Your final score on the survey part will be uploaded manually later on in the semester.

Make sure to check this out to make sure your group is recorded correctly

Assigning everyone the highest score will not fetch you 100% score.

Cut Property Lemma for MSTs

Condition: S and V\S are non-empty

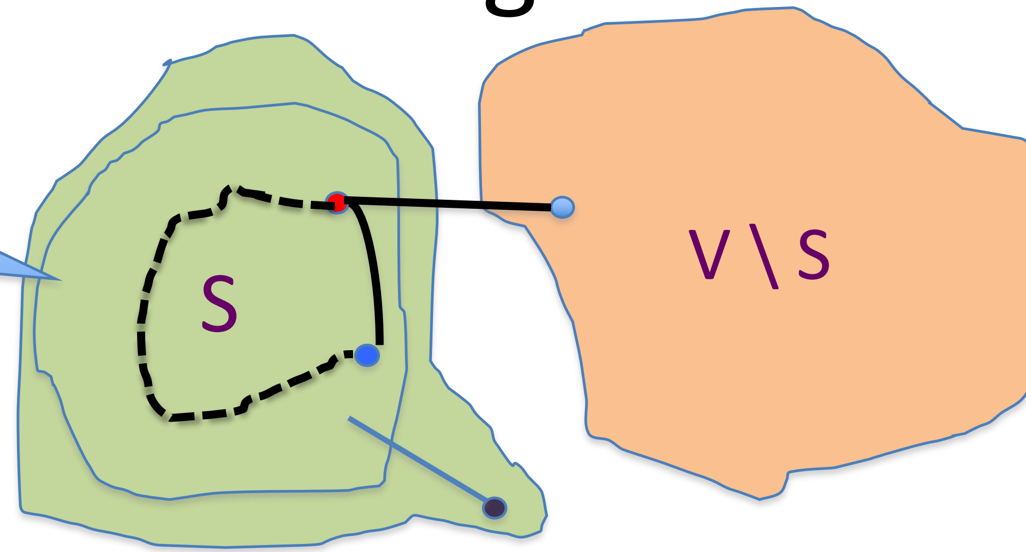


Cheapest crossing edge is in **all** MSTs

Assumption: All edge costs are distinct

Optimality of Kruskal's Algorithm

Nodes connected to red in (V,T)



Input: $G=(V,E)$, $c_e > 0$ for every e in E

$T = \emptyset$

Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

S is non-empty

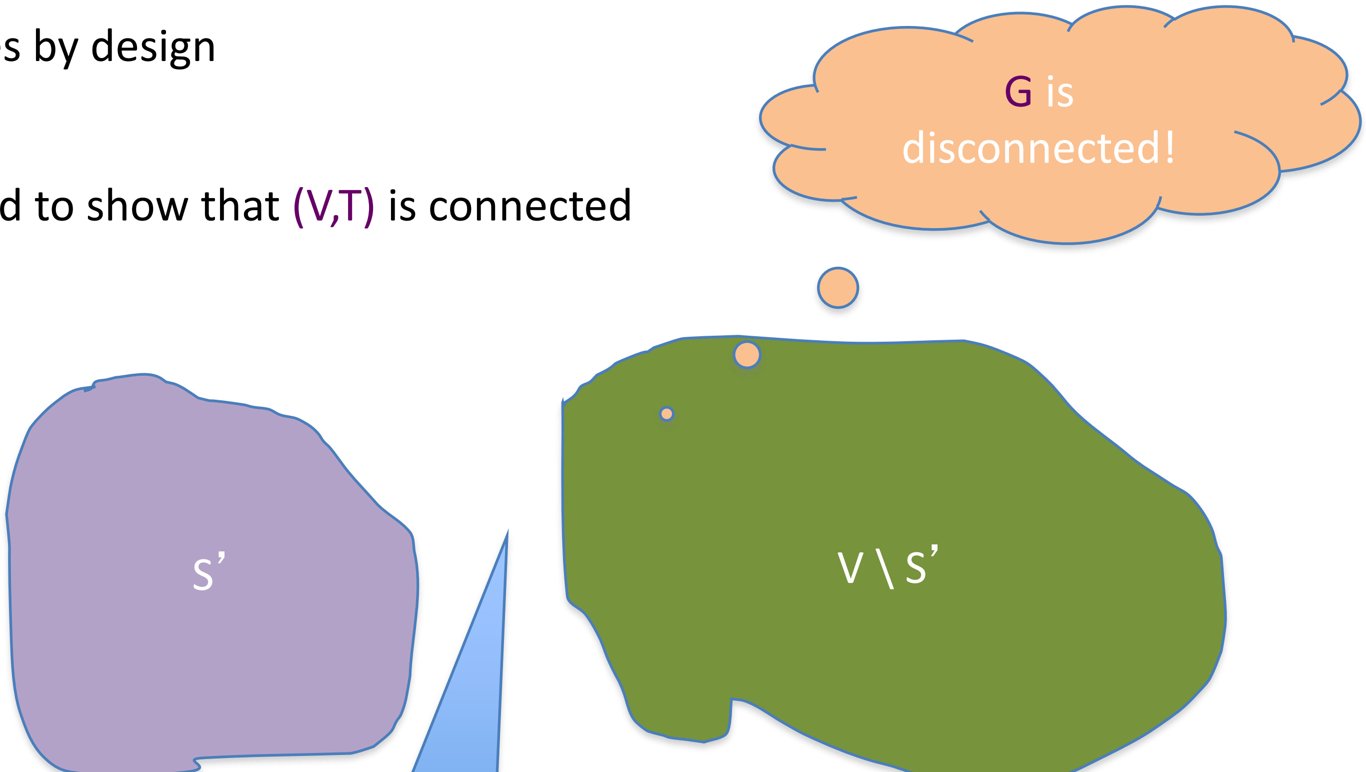
V\S is non-empty

First crossing edge considered

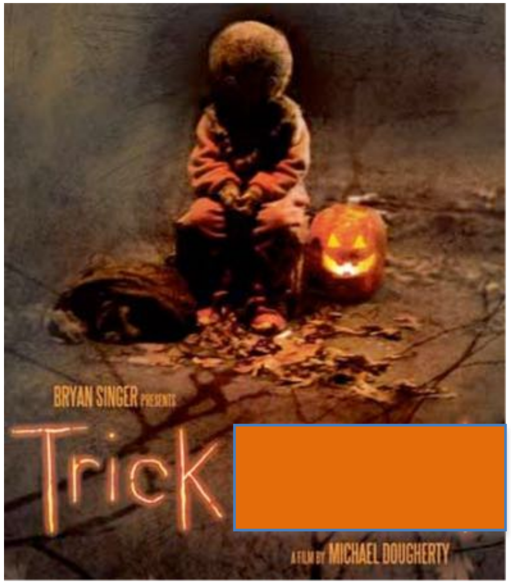
Is (V,T) a spanning tree?

No cycles by design

Just need to show that (V,T) is connected



Removing distinct cost assumption



Change all edge weights by very small amounts

Make sure that all edge weights are distinct

MST for "perturbed" weights is the same as for original

Changes have to be small enough so that this holds

EXERCISE: Figure out how to change costs

Running time for Prim's algorithm

Similar to Dijkstra's algorithm

O(m log n)

Input: G=(V,E), c_e> 0 for every e in E

S = {s}, T = Ø

While S is not the same as V

Among edges e= (u,w) with u in S and w not in S, pick one with minimum cost

Add w to S, e to T

Running time for Kruskal's Algorithm

Can be implemented in $O(m \log n)$ time (Union-find DS)

Input: $G=(V,E)$, $c_e > 0$ for every e in E

$T = \emptyset$

Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

$O(m^2)$ time overall

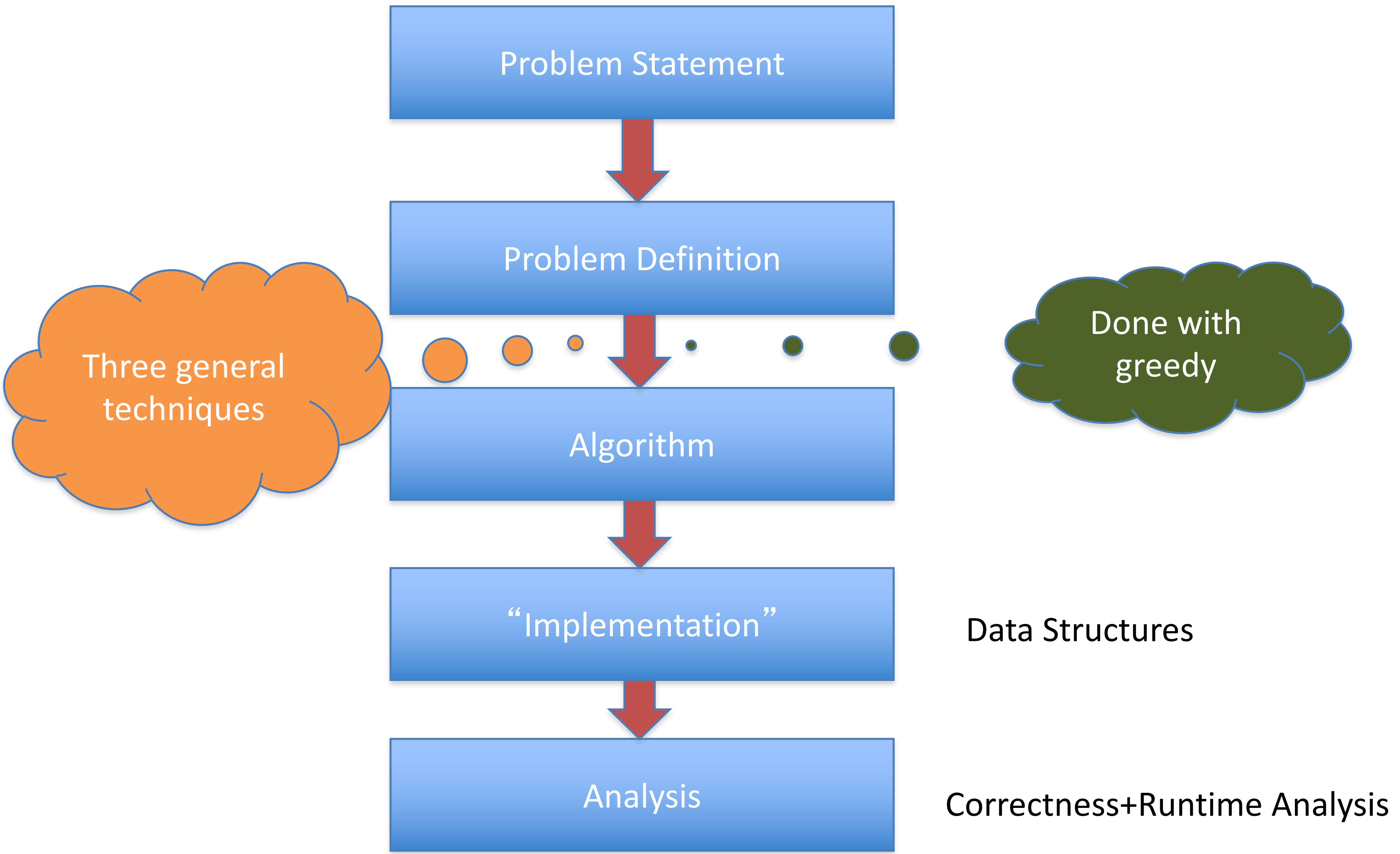
Can be verified in $O(m+n)$ time

Joseph B. Kruskal

Reading Assignment

Sec 4.5, 4.6 of [KT]

High Level view of the course



Trivia

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

"Patch up" the solutions to the sub-problems for the final solution

Sorting

Given n numbers order them from smallest to largest

Works for any set of elements on which there is a total order

Insertion Sort

Input: $a_1, a_2, \ldots, a_n$

Output: $b_1, b_2, \ldots, b_n$

$O(n^2)$ overall

Make sure that all the processed numbers are sorted

$b_1 = a_1$

for $i = 2 \ldots n$

Find $1 \le j \le i$ s.t. a_i lies between b_{j-1} and b_j

Move b_j to b_{i-1} one cell "down"

$b_j = a_i$

O(log n)

O(n)

a	b
4	[illegible]
3	[illegible]
2	[illegible]
1	4

Other $O(n^2)$ sorting algorithms

Selection Sort: In every round pick the min among remaining numbers

Bubble sort: The smallest number “bubbles” up

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

"Patch up" the solutions to the sub-problems for the final solution

Mergesort Algorithm

Divide up the numbers in the middle

Unless n=2

Sort each half recursively

Merge the two sorted halves into one sorted output

How fast can sorted arrays be merged?

Mergesort algorithm

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

MergeSort(a, n)

If $n = 1$ **return** the order a_1

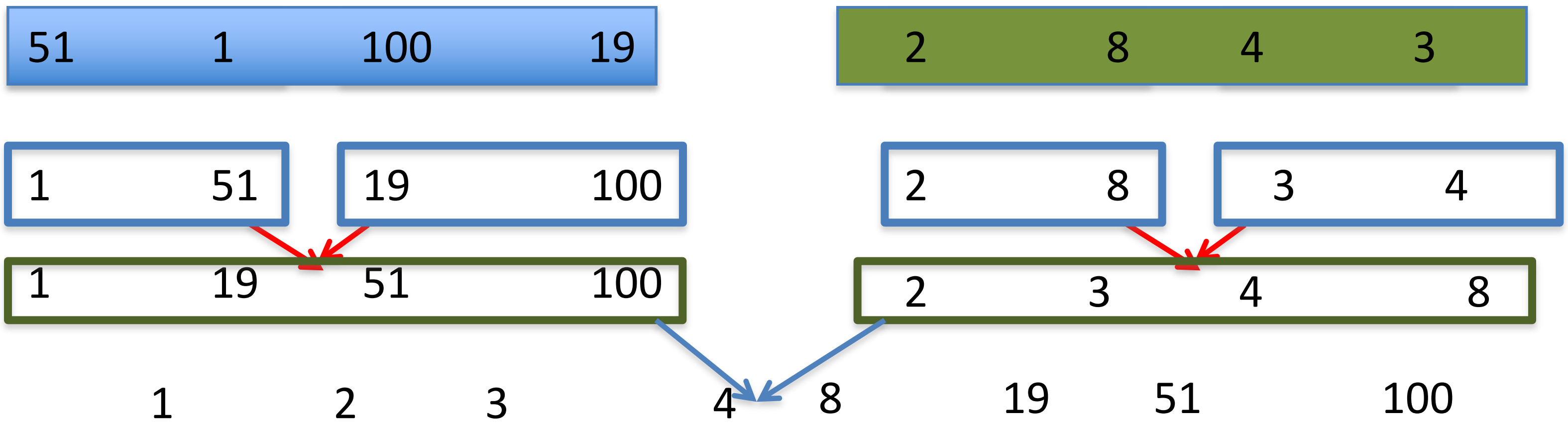
If $n = 2$ **return** the order $\min(a_1,a_2); \max(a_1,a_2)$

$a_L = a_1,\ldots, a_{n/2}$

$a_R = a_{n/2+1},\ldots, a_n$

return MERGE (MergeSort(a_L, $n/2$), MergeSort(a_R, $n/2$))

An example run



MergeSort(a, n)

If $n = 1$ **return** the order a_1

If $n = 2$ **return** the order $\min(a_1,a_2); \max(a_1,a_2)$

$a_L = a_1, \ldots, a_{n/2}$

$a_R = a_{n/2+1}, \ldots, a_n$

return MERGE (MergeSort(a_L, $n/2$), MergeSort(a_R, $n/2$))

Correctness

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

MergeSort(a, n)

If n = 1 **return** the order a_1
If n = 2 **return** the order $\min(a_1,a_2); \max(a_1,a_2)$

$a_L = a_1,\ldots, a_{n/2}$

$a_R = a_{n/2+1},\ldots, a_n$

return MERGE (MergeSort(a_L, n/2), MergeSort(a_R, n/2))

By induction on n

Inductive step follows from correctness of MERGE