Lecture 23

CSE 331

Oct 27, 2023

Project deadlines coming up

Tue, Oct 31		(**HW 5 in**)
Wed, Nov 1	Multiplying large integers F22 F21 F19 F18 F17 x^2	[KT, Sec 5.5] *Reading Assignment:* Unraveling the mystery behind the identity
Fri, Nov 3	Closest Pair of Points F22 F21 F19 F18 F17 x^2	[KT, Sec 5.4] (**Project (Problems 1 & 2 `Coding`) in**)
Mon, Nov 6	Kickass Property Lemma F22 F21 F19 F18 F17 x^2	[KT, Sec 5.4] (**Project (Problems 1 & 2 `Reflection`) in**)

Group formation instructions

Autolab group submission for CSE 331 Project

The lowdown on submitting your project (especially the coding and reflection) problems as a group on Autolab.

Follow instructions **EXACTLY** as they are stated

The instruction below are for Coding Problem 1

You will have to repeat the instructions below for EACH coding AND reflection problem on project on Autolab (with the appropriate changes to the actual problem).

Form your group on Autolab

Groups on Autolab will NOT be automatically created

You will have to form a group on Autolab by yourself (as a group). Read on for instructions on how to go about this.

Click to add notes

Do not use Join a Group “feature”

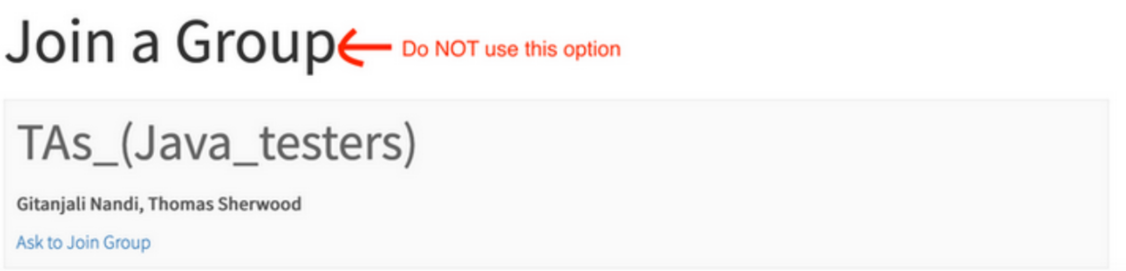
note @331

stop following **143 views**

Actions

When forming groups on Autolab

I forgot to add this warning earlier but I have updated the Autolab project page to add a warning to **not click on the "Join a Group" function when you are creating your group on Autolab:**

> **Do NOT click on `Join a Group`**
>
> **Do NOT use the "Join a Group" feature. ONLY follow the instructions above EXACTLY.**
>
> This step can be un-done but needs intervention on our part BUT that'll cause delays on your side and we are not responsible if you miss your deadline due to this delay.
>
> Here is what such an option looks like (the actual group name and group members would be different in your case:
>
> 
>

project autolab

Please be in touch w/ your group

note @321 stop following **153 views**

Actions

Please respond to your project group mates

Please do respond back if a group project member reaches out to you to get started on the project. Just FYI, ***I always reserve the right to kick you out of your group (which means a 0 for you)*** **in case you are unresponsive to repeated requests from your group members**.

I understand some of you might be busy now-- it is fine with me if your group decide *as a whole* how the work will be divided (so e.g. someone does less work on the initial problems and someone does more work on the later problems). As long as the group agrees, I do not care about the details.

But please do respond back in a timely fashion: not doing so is you not doing your part in a group project.

project

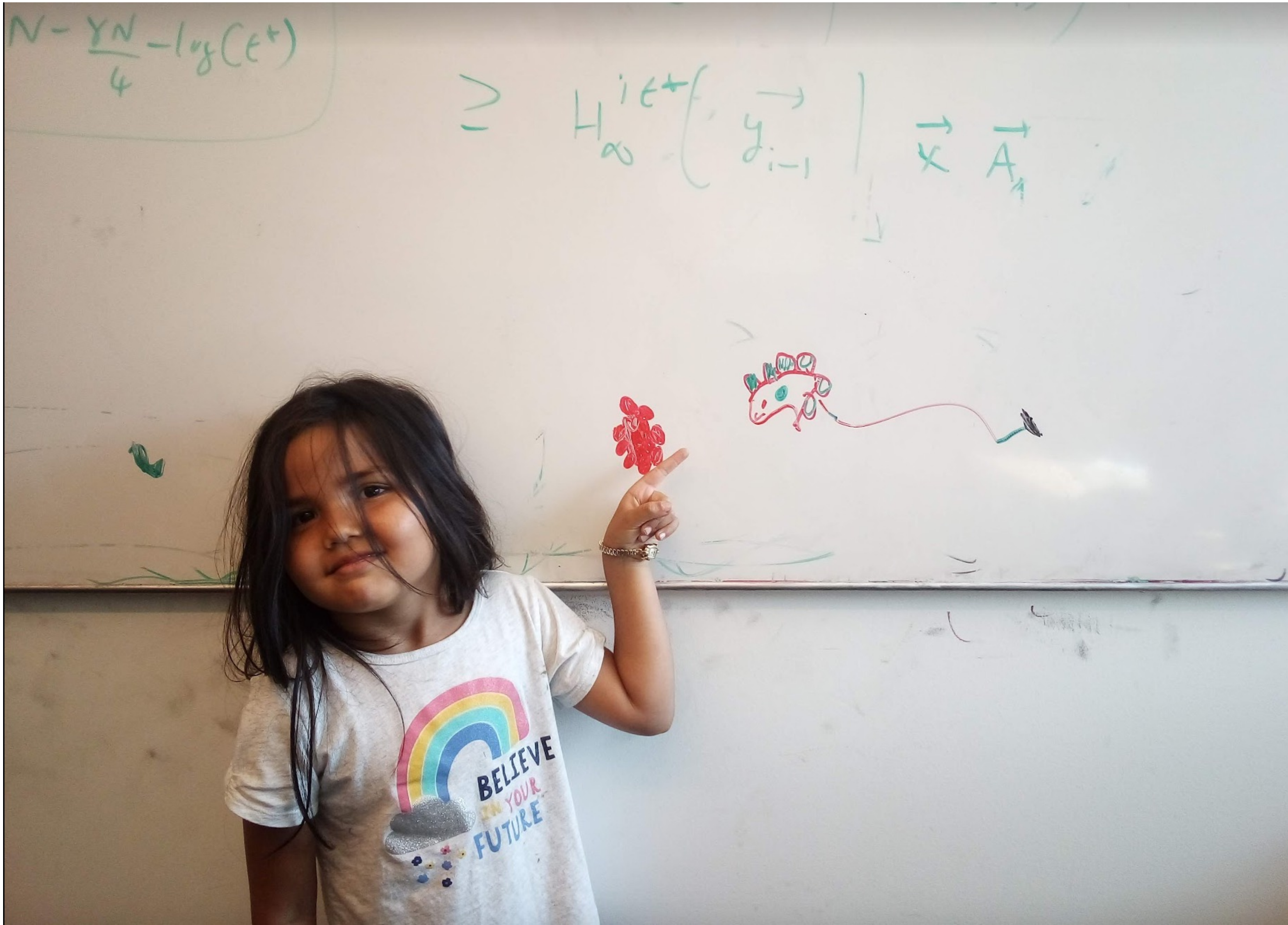
Edit good note | 0 Updated 2 weeks ago by Atri Rudra

Temp mid-sem grade + 1-on-1 slots

Hopefully by Sat

By Sun at the latest

Questions/Comments?



Kruskal's Algorithm

Joseph B. Kruskal

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

$T = \emptyset$

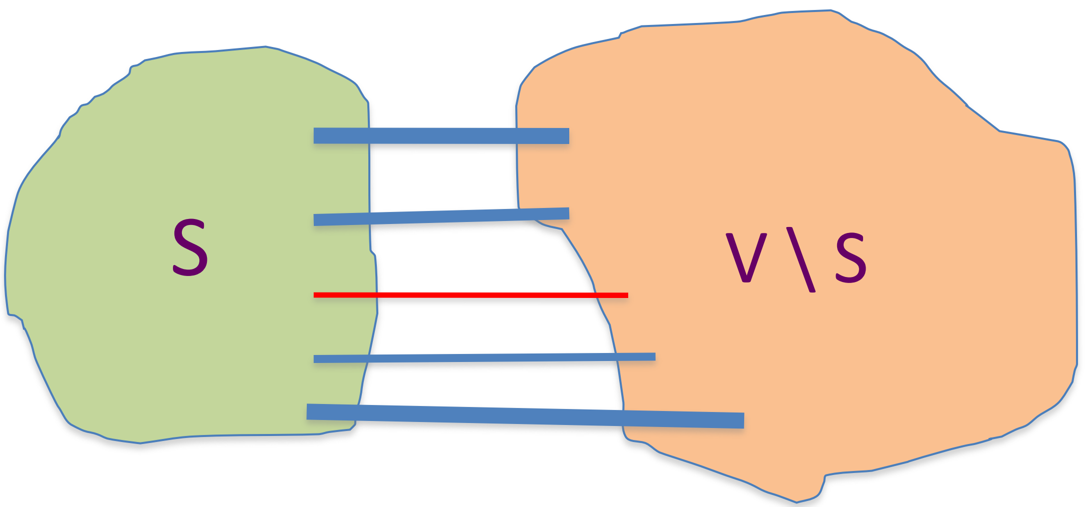
Sort edges in increasing order of their cost

Consider edges in sorted order

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

Cut Property Lemma for MSTs

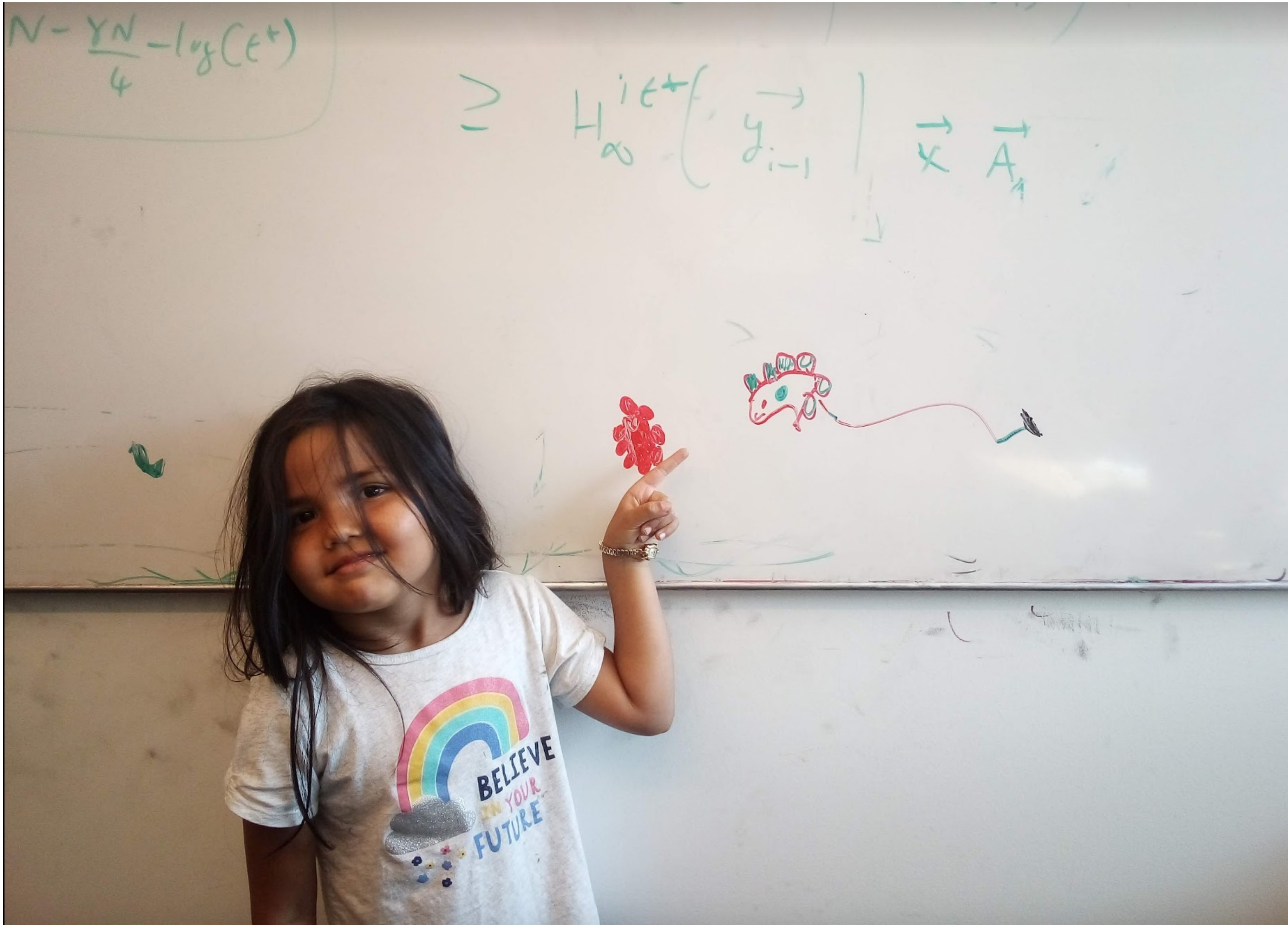
Condition: S and V\S are non-empty



Cheapest crossing edge is in **all** MSTs

Assumption: All edge costs are distinct

Questions/Comments?



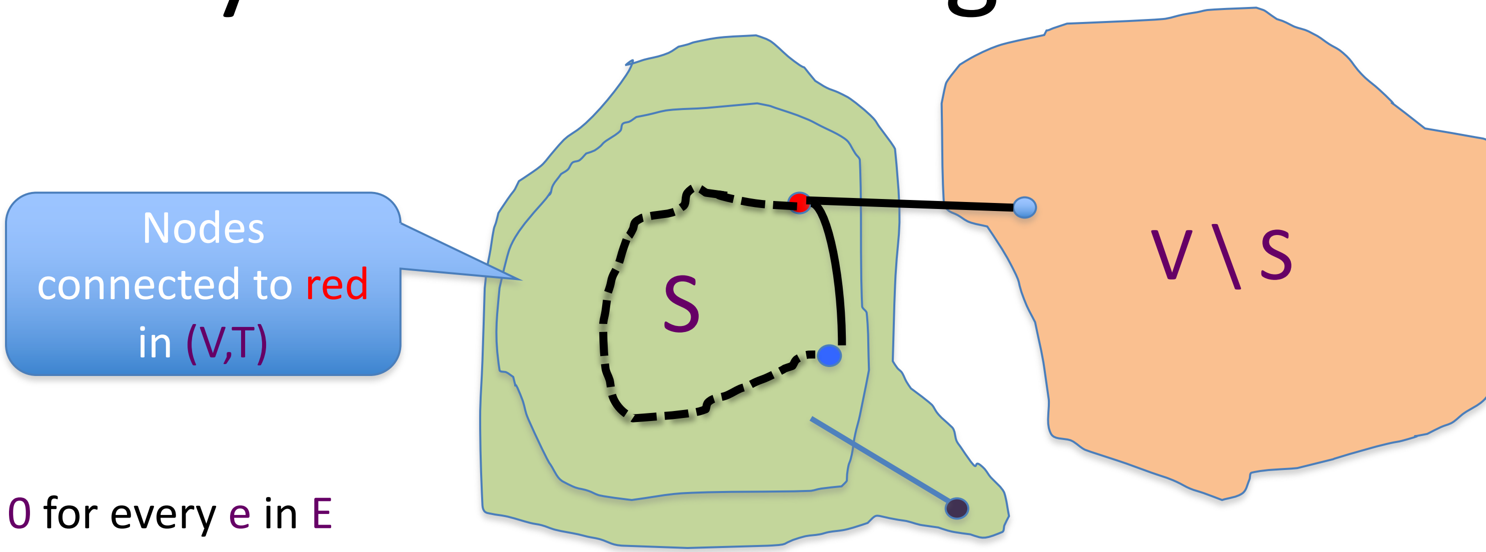
Today's agenda

Optimality of Kruskal's algorithm

Remove distinct edge weights assumption

Quick runtime analysis of Prim's+Kruskal's

Optimality of Kruskal's Algorithm



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

T = Ø

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

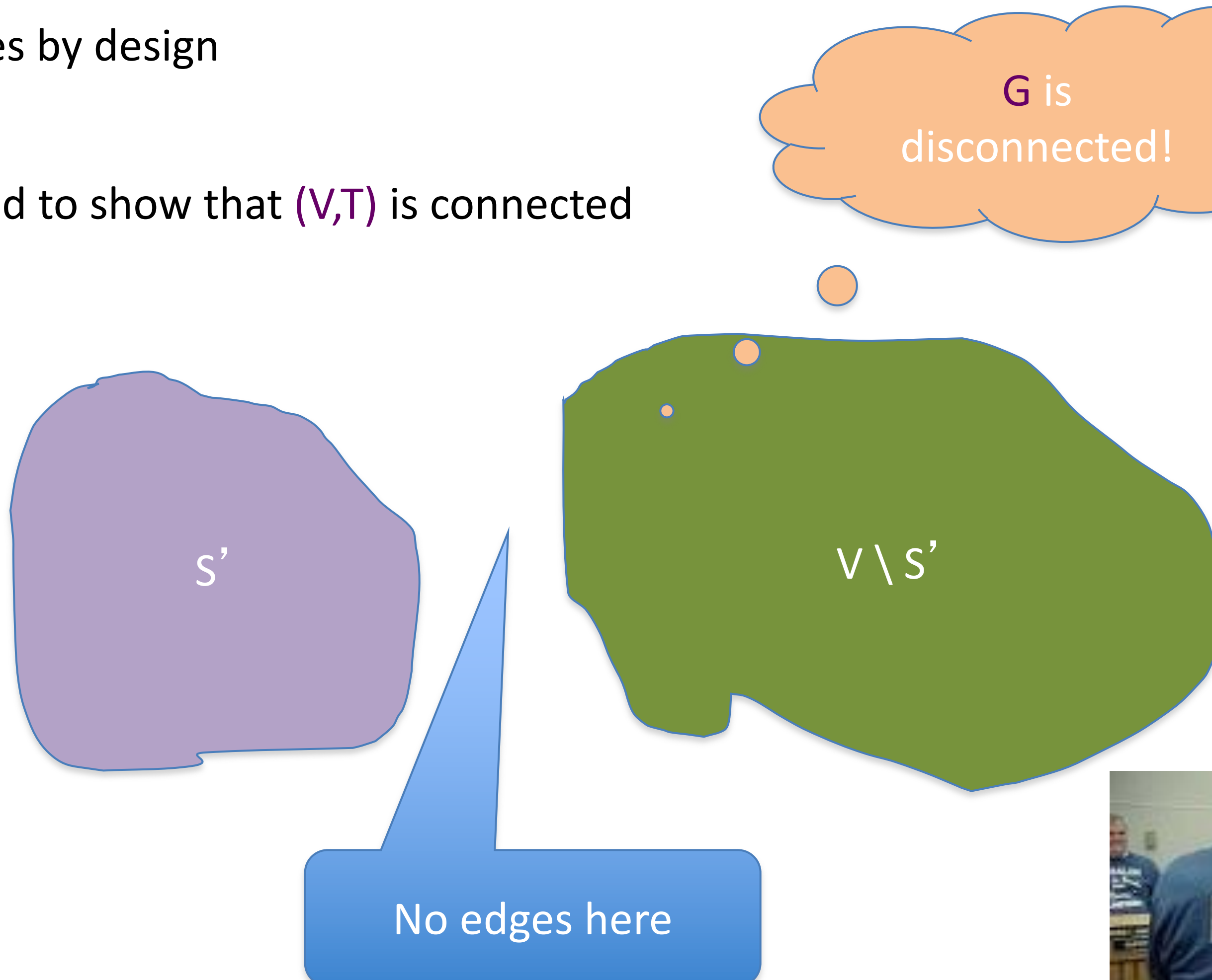
V\S is non-empty

First crossing edge considered

Is (V,T) a spanning tree?

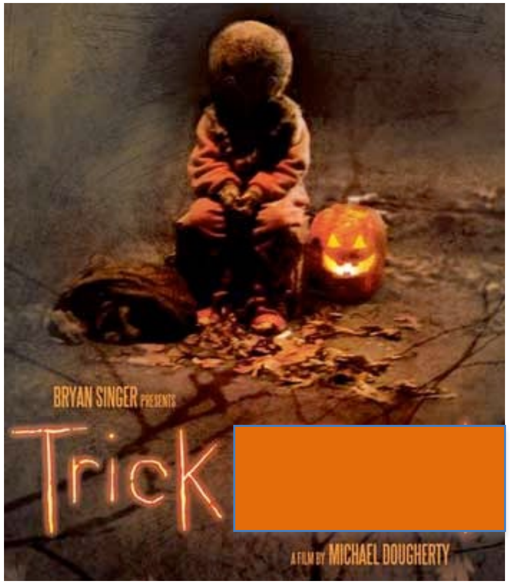
No cycles by design

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



Removing distinct cost assumption

Change all edge weights by very small amounts



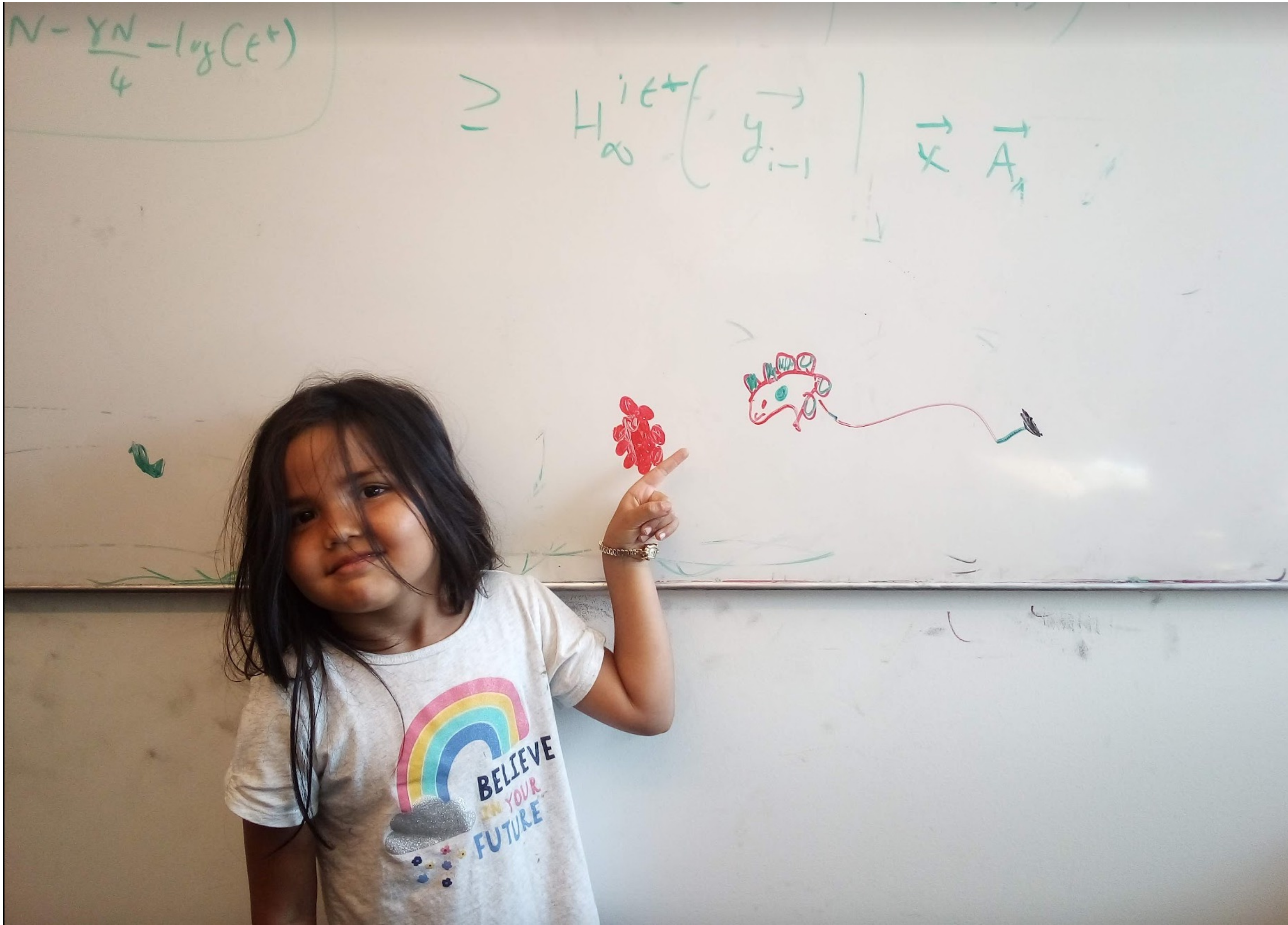
Make sure that all edge weights are distinct

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

Questions/Comments?



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

$S = \{s\}$, $T = \emptyset$

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

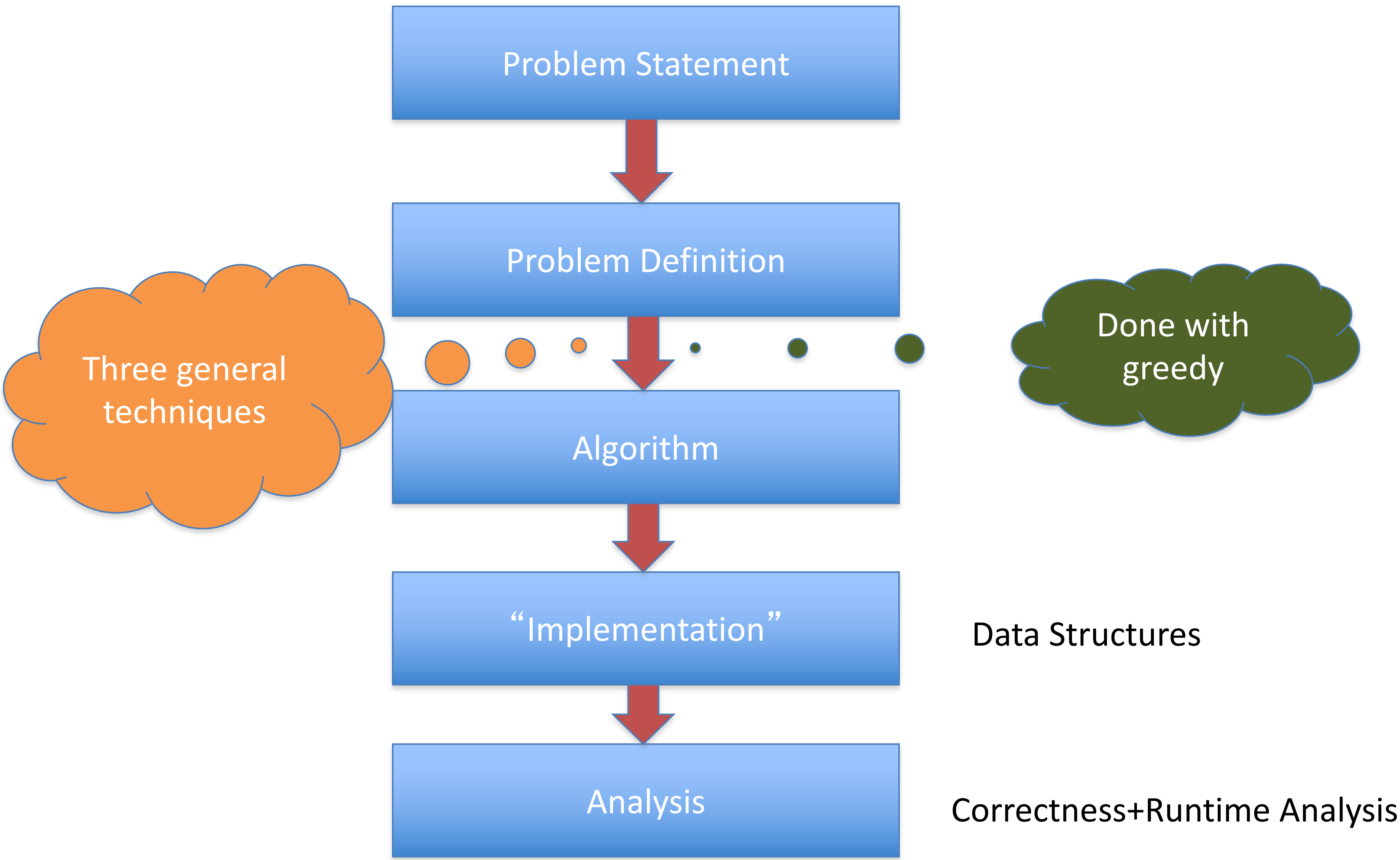
Can be verified in $O(m+n)$ time

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: $b_1, b_2, \ldots, b_n$

$b_1 = a_1$

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"

  $b_j = a_i$

$O(n^2)$ overall

$O(\log n)$

$O(n)$

Make sure that all the processed numbers are sorted

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?

Group talk time

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

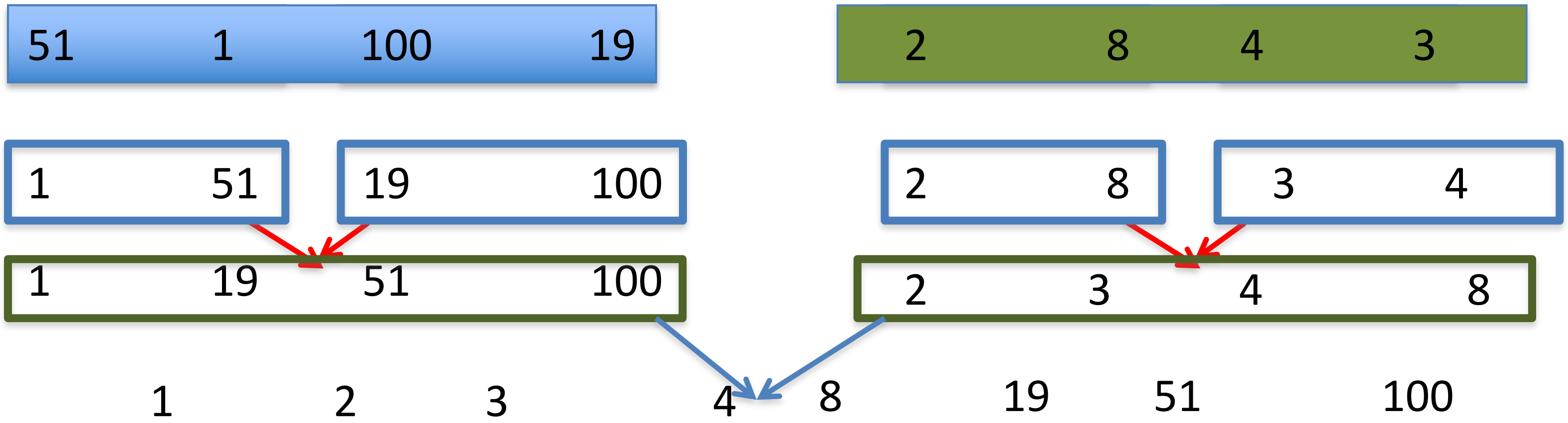
If $n = 2$ **return** the order $\min(a_1,a_2); \max(a_1,a_2)$

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

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

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

Runtime analysis on the board…