WEEK8:
GRAPH ALGORITHMS 2

Agenda

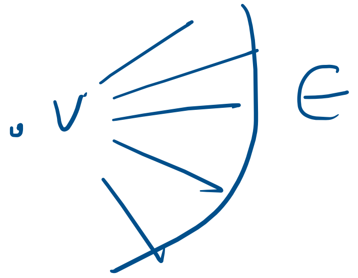
Survey Questions:

- Prim’s Algorithm Time Complexity
- Kruskal’s Algorithm Time Complexity
- How can we tell if a greedy algorithm will work on a problem and how can we implement Greedy algorithms.
- Will we have our midterm grades in week 8?

Prim's Algorithm (Naïve) Time Complexity

```
def prims(G):
    Result = {}
    visited = {} #pick one vertex from V

    while(len(visited)<V):
        find (a,b) where
            (a is in visited and b is not in visited) and (Edge(a,b) is min)

        Result.add((a,b))
        visited.add(b)

    return Result
```

V



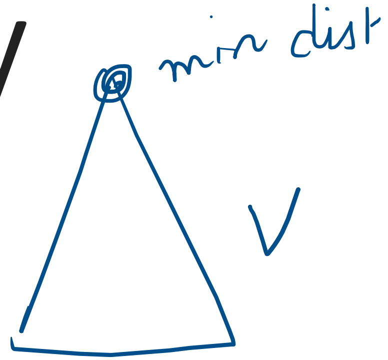
E

O(VE)

Prim’s Algorithm Time Complexity

```
def prims(G):
   s <- pick a source vertex from V
   for v in V:
      dist[v] = infinity
      prev[v] = Empty
   #initalize source
   dist[v] = 0
   prev[v] = s
   #update neighbouring nodes of s
   for node in s.neighbours
      dist[v] = w(s,node)
      prev[v] = s

   while(len(visited)<len(V)):
      CurrentNode = unvisited vertex v with smallest dist[v]
      MST.add((prevNode, CurrentNode))
      for node in CurrentNode.neighbours:
         dist[node] = min(w(CurrentNode, node), dist[node])
         if dist[node] updated: prev[node] = CurrentNode
      visited.add(CurrentNode)
   return MST
```

A	∞
B	$\cancel{\infty}$
C	$\cancel{\infty}$



V

$$O\left((\log V) \times V\right)$$

$+$

$$E(\log V)$$

$$O \approx (E \log V)$$

$E \log v$ vs $E \log E$

$A \approx B$

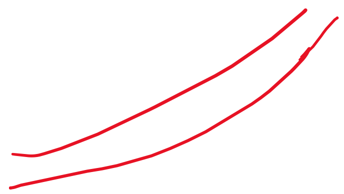
$\log v$ vs $\log E$

$E = v^2$

$E \log v^2$

$\cancel{2} E \log v$

$E \log v$



kruskal’s **Algorithm Time Complexity**

The idea behind Kruskal’s

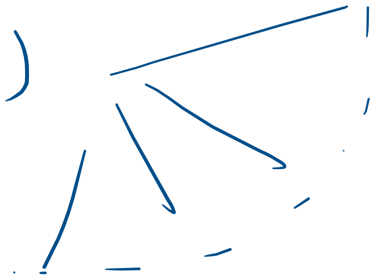
```
def Kruskal(V,E):
   sorted_E = sort E by increasing weight
   MST ={}

   for e in sorted_E:
      if MST and e don't cycles:
         add e to MST
   return MST
```

$O(E \log E)$

$+$

E, V: $O(EV)$

(u,v)



$O(EV)$

kruskal's **Algorithm Time Complexity**

Using Disjoint Set

```
def Kruskal(V,E):
    E_sorted = sort E by increasing weight

    for v in V:
        make_set(v)
    msv = {}

    for (u,v) in E_sorted:
        if(Find_set(u) != Find_set(v)):
            MST.add((u,v))
            Union(u,v)
    return MST
```

```
def Kruskal(V,E):
    sort E by increasing weight          — E log E
                                           +
    for v in V:
        create a tree for each V         — V
                                           +
    MST = {}
                                           +
    for i in Range(|E|):
        (u,v) <- lightest edge in E      — E
                                           +
    if u and v not in same tree:
        MST.add((u,v))
        merge u and v trees              — O(E(log V))

    return MST
```

$O(E \log E)$

Other Questions:

- How can we tell if a greedy algorithm will work on a problem and how can we implement Greedy algorithms.

  1. **Greedy Choice Property**: A globally optimal solution can be obtained by making a locally optimal choice. That is, for sub-problems, if we make the best possible choice (locally greedy choice) this would result in the optimal solution for the bigger problem (Global optimal solution).
  2. **Optimal Substructure**: We have seen this term in the dynamic programming section. A problem is said to have an optimal substructure if the optimal solution for the problem can be obtained by taking the optimal solution for the sub-problems.
  3. See if you can come up with counter example

- midterm grades?