Final review

Disjoint set

Definition

a set of elements partitioned into a number of disjoint subsets.

Implementation

Use an array to store each element's parent. This forms a special tree, where children points to the parents.

- Init:

```
for (int i = 0; i < n; i++>) {
  parent[i] = i;
}
```

- Find: Find the root element for i

```
size_t find(size_t i) {
  while(parent[i] != i) {
    i = parent[i];
  }
  return i;
}
```

$$T_{find} = O(h)$$

- set_union: join two set into one set

```
void set_union(size_t i, size_t j) {
  i = find(i);
  j = find(j);
  if (i != j) {
    parent[j] = i;  // merge j into i
  }
}
```

Optimization

Union by height

Point the root of shorter tree to the root of the taller tree. The height will increase iff. both tree are of the same height.

Worst case: Binomial coefficients.

$$h = O(\log n)$$

Path Compression

```
size_t find(size_t i) {
  if(parent[i] == i) {
    return i;
  } else {
    parent[i] = find(parent[i]);
    return parent[i];
  }
  return i;
}
```

Next call to find will become $\Theta(1)$. Very useful when find operation is frequent.
This cost $O(h)$ memory.

Amortized time complexity

$$O(\alpha(n))$$

where $\alpha(n)$ is the inverse of Ackermann function $A(i, i)$.
In practice, $\alpha(n) \leq 4$ is small enough, but in theory, it's still a function of n.

Graph & Graph traversal

- graph, directed/undirected graph,
- path, simple path, simple cycle
- connectedness
- weighted graph
- forest
- **NOTE** without specification, consider simple path (no duplicated path, no self-loop)

Undirected graph

Graph witout direction on edge.

$$|E| \leq O\left(|V|^2\right)$$

- degree: the count edges connected to one noede.

Directed graph

In a directed graph, the edges on a graph are be associated with a direction.

$$|E| \leq O\left(|V|^2\right)$$

- in/out degree
- sink/source
- strongly/weakly connected
- weighted directed graphs

Representation

Adjacency matrix

- $O(|V|^2)$ memory
- determine if v_j is adjacency to v_k is $O(1)$
- finding all neighbors of v_j is $\Theta(|V|)$

Adjacency list

- Requires $\Theta(|V|+|E|)$ memory
- On average: determine if v_j is adjacency to v_k is $O(|E|/|V|)$
- On average: finding all neighbors of v_j is $\Theta(|E|/|V|)$

Bread-first traversal

- use a queue
- The size of the queue is $O(|V|)$
- Time $O(|V|+|E|)$

Depth-first traversal

- use stack
- At most $O(|V|)$ elements in the stack (on a path)
- Time $O(|V|+|E|)$

NPC

- Polynomial reduction
- P, NP, NP-complete, NP-hard

Common NPC problem

1. SAT, 3-SAT
2. VERTEX-COVER
3. INDEPENDENT-SET
4. SET-COVER
5. 3-COLOR
6. Knapsack

Give a problem A to be proved, and another NP-complete problem B.

1. Prove A is in NP: Construct a polynomial verifier.
   1.1. You have to specify what is a certificate.
   1.2. You should explain why it is in polynomial time briefly. (You don't have to give a complete algorithm)
2. Prove that $B \leq_P A$ by giving a polynomial-time reduction from an arbitary instance in B to the instance in A.
   2.1. You have to explain why the construction is in polynomial time briefly.
3. Prove the correctness of the construction. The key here is to prove that α is a 'yes-instance' in A iff. β is a 'yes-instance' in B where β is the instance construct from α. (In two direction)