Week 11 Summary: Greedy Exchange Proofs and Minimum Spanning Trees

Z5007 Programming and Data Structures · MTech DSAI · Odd Semester 2026-27 · Dr. Innocent Nyalala

1. The Greedy Design Pattern and Exchange Arguments

A greedy algorithm makes the locally optimal choice at each step without revising earlier decisions. Correctness relies on two properties:

1. **Greedy Choice Property:** A globally optimal solution exists that includes the first greedy step.
2. **Exchange Argument:** Given any optimal solution differing from the greedy one, swapping the greedy choice in preserves feasibility without worsening cost.

Classic examples: Earliest-finish-time interval scheduling maximizes non-overlapping events in $O(n \log n)$, and sorting items by v_i/w_i solves Fractional Knapsack optimally in $O(n \log n)$.

2. The Cut Property for Minimum Spanning Trees (MST)

For a connected undirected weighted graph $G = (V, E)$, a spanning tree connects all $|V|$ vertices using $|V| - 1$ edges with minimum total weight. For any cut $(S, V \setminus S)$ that no chosen MST edge crosses yet, the minimum-weight edge crossing that cut is guaranteed to belong to some MST of G.

3. Kruskal's Algorithm vs. Prim's Algorithm

Algorithm	Strategy	Data Structure	Complexity
Kruskal	Sort all edges ascending; add if endpoints disconnected	Disjoint Set Union (path comp. + rank)	$O(\|E\| \log \|E\|)$
Prim	Grow one tree from s; pick lightest frontier edge	Binary Min-Heap	$O((\|V\| + \|E\|) \log \|V\|)$

Disjoint Set Union (Union-Find) Complexity

Combining *union by rank* with *path compression* in `find(x)` reduces amortized cost per operation to $O(\alpha(n))$, where $\alpha(n) \leq 4$ for every practical input size.

4. Worked Check

On vertices $\{A, B, C, D, E\}$ with edges $(A, B, 1)$, $(B, C, 2)$, $(D, E, 2)$, $(C, D, 3)$, $(A, C, 4)$, $(C, E, 5)$, $(B, D, 6)$, Kruskal inspects weights in order $1, 2, 2, 3$ and accepts $(A, B), (B, C), (D, E), (C, D)$ for total MST weight 8, skipping (A, C) because A and C already share a component.