Computational Discrete Optimization

CO 353

Chaitanya Swamy

LaTeXed by *Sibelius Peng*

Preface

Disclaimer Much of the information on this set of notes is transcribed directly/indirectly from the lectures of CO 353 during Winter 2022 as well as other related resources. I do not make any warranties about the completeness, reliability and accuracy of this set of notes. Use at your own risk.

Discrete optimization problems are underlying decisions that have a discrete flavor, e.g., YES/NO or $\{0, 1\}$ decisions.

The focus in this course will be on algorithms, modelling. Broad classes of problems that we will study are network connectivity problems, location problems, general integer programs.

For any questions, send me an email via `https://notes.sibeliusp.com/contact`.

You can find my notes for other courses on `https://notes.sibeliusp.com/`.

Sibelius Peng

Contents

1

Graph Algorithms

1.1 Definitions, Notations & Terminology

A **graph** is a tuple (V, E), where V is set of **nodes/vertices**, E is set of **edges**, where edges **joins** two nodes.

If e is an edge that joins nodes u, v, then we denote this by $e = uv$. u, v are called **ends** of e. e is **incident** to nodes u, v. We are not allowing parallel edges, i.e., $e = uv$, and $e' = u'v'$ are distinct edges, then $\{u, v\} \neq \{u', v'\}$.

An u-v path in $G = (V, E)$ where $u, v \in V, u \neq v$, is a sequence of nodes $u_1 = u, u_2, \ldots, u_k, u_{k+1} = v$, where $u_i u_{i+1} \in E \; \forall i = 1, \ldots, k$. A **cycle** in G is a sequence of nodes $u_1, u_2, \ldots, u_k, u_{k+1} = u_1$ where $u_i u_{i+1} \in E \; \forall i = 1, \ldots, k$, and u_i's are distinct. Since there are no parallel edges, we can also identify a path/cycle by its sequence of $u_i u_{i+1}$ edges. So we will often refer to a path/cycle as a set of edges.

A graph G is **connected** if it has a $u - v$ path $\forall u, v \in V$ $(u \neq v)$. G is acyclic if G does not have a cycle. A **tree** is a connected, acyclic graph.

Let $G = (V, E)$ be a connected graph, and $T = (V_T, E_T)$ be a tree. IF $E_T \subseteq E$ and $V_T = V$, then we say that T is a **spanning tree** of G.

If C is a cycle, and $e \in C$, then $C - \{e\}$ still connects all nodes of C. So if G is a connected graph, and it contains a cycle C, and $e \in C$, then $G - \{e\} := (V, E - \{e\})$ is a connected graph. Hence, a spanning tree of G is a minimal connected subgraph of G. I.e., if $T = (V, F)$ where $F \subseteq E$ is a minimal set such that (V, F) is connected, then T is a spanning tree of G. If $T = (V, F)$ contains a cycle, then F is not minimal.

In **directed graph**, each edge has a direction, and goes **from** a node **to** another node.

1.2 Shortest paths: Dijkstra's algorithm

Problem Given a directed graph $G = (V, E)$ with edge costs $\{c_e \geq 0\}$ and a node $s \in V$, find the shortest path from s to all other nodes. The "shortest" path means path with the smallest total edge cost under the c_e edge costs.

Notation For a path P, let