Advanced Algorithms Notes

Protopapa Francesco

2nd Semester 2022/23

Graphs basics

Definitions

- Given an edge $e = (u, v)$
  - e is **incident** on u and v
  - u and v are **adjent**
- **neighbours of a vertex v**: all vertices u s.t. $(u, v) \in E$
- **degree of a vertex**: $d(v)$ is the number of edges incident on v
- **simple path**: $u_1, u_2, ..., u_k$ all distinct and $\forall 1 \leq i < k. (u_i, u_{i+1}) \in E$
- **cycle**: symple path s.t. $u_1 = u_k$
- **subgraph** (of a graph $G = (V, E)$): $G' = (V', E')$ s.t. $V' \subseteq V, E' \subseteq E$ and the edges of E' are incident only on V'
- **spanning subgraph**: a subgraph with $V' = V$
- **connected graph**: if $\forall u, v \in V. \exists$ a path from u to v
- **connected components**: a partition of G in subgraphs $\forall 1 \leq i \leq k. G_i = (V_i, E_i)$ s.t.
  - $\forall i. G_i$ is connected
  - $V = V_1 \cup V_2 \cup ... \cup V_i$
  - $E = E_1 \cup E_2 \cup ... \cup E_i$
  - $\forall i \neq j$ there is no edge between i and j
- **tree**: connected graph without cycles
- **forest**: set of trees (disjoint)
- **spanning tree**: spanning subgraph, connected and without cycles
- **sapnning forest**: spanning subgraph without cycles
- **size of a graph**: $n + m$ where $n = |N|$ and $m = |E|$
- **minimum spanning tree**: a spanning tree T of G s.t. $w(T) = \sum_{(u,v) \in T} w(u, v)$ is minimized
- **cut** (of a graph $G = (V, E)$): is a partition of $V \to (S, V \setminus S)$
- an edge $(u, v) \in E$ **crosses** the cut if $u \in S$ and $v \in V \setminus S$ (or viceversa)
- a cut **respects** a set of edges A if no edges of A crosses the cut
- given a cut, an edge that crosses the cut and is of minimum weight is called **light edge**
- given a weighted graph, the **length of a path** $P = v_1, v_2, ..., v_k$ is defined as $\text{len}(P) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$

- a **shortest path** from a vertex u to a vertex v is a path with minimum length among all $u - v$ paths
- the **distance** between two vertices s and t, denoted $\text{dist}(s, t)$ is the length of a shortest path from s to t, if there is no path at all from s to t then $\text{dist}(s, t) = +\infty$
- a **flow network** is a directed graph $G = (V, E)$ where each edge has a **capacity** $c(e) \in \mathbb{R}^+$, along with a designated **source** $s \in V$ and **sink** $t \in V$ (for convience $c(e) = 0$ if $e \notin E$)
- a **flow** is a function $f : E \to \mathbb{R}^+$ satifying the following constraints:
  1. (capacity) $\forall e \in E.\, f(e) \leq c(e)$
  2. (conservation) $\forall u \in V \setminus \{s, t\}$ we have
  $$\sum_{v \in V \ | \ (v,u) \in E} f(v, u) = \sum_{v \in V \ | \ (u,v) \in E} f(u, v)$$
- the value of a **flow** is
$$|f| = \sum_{v \in V \ | \ (s,v) \in E} f(s, v)$$
- given a flow network G and a flow f, the **residual network** of G (w.r.t flow f), G_f, is a network with vertex set V and with edge set E_r as follows:
  - if $f(e) < c(e)$, add e to G_f with capacity $c_f(e) = c(e) - f(e)$
  - if $f(e) > 0$, add another edge (v, u) to G_f with capacity $c_f(e) = f(e)$
- a **tour** is a cycle that visits every vertex exactly once
- a **clique** is another name for complete graph
- a **vertex cover** of a graph is a set of vertices that includes at least one endpoint of every edge of the graph
- a set of edges A is a **matching** if $\forall e, e' \in A.\, e \cap e' = \emptyset$
- a **maximal matching** A is such that $\forall e \in E.\, A \cup \{e\}$ is not a matching
- given a tree, a **full preorder chain** is a list with repetitions of vertices of the tree which identifies the vertices reached from the recursive calls of $\text{PREORDER}(T, v)$
- a path or a cycle is **Eulerian** if it visits avery edge of the graph exactly once
- a connected graph is **Eulerian** if exists an Eulerian cycle
- a **multi-graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ s.t. $\mathcal{V} \subseteq \mathbb{N}$ and $\mathcal{E}$ is a multiset of elements (u, v) s.t. $u \neq v$
- given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ connected, a **cut** $\mathcal{C} \subseteq \mathcal{E}$ is a multiset of edges s.t. $\mathcal{G}' = (\mathcal{V}, \mathcal{E} \setminus \mathcal{C})$ is not connected
- given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $e = (u, v) \in \mathcal{E}$, the **contraction of $\mathcal{G}$ with respect to e**, $\mathcal{G}_{/e} = (\mathcal{V}', \mathcal{E}')$, is the multigraph with:
  - $\mathcal{V}' = \mathcal{V} \setminus \{u, v\} \cup \left\{z_{u,v}\right\}\left(z_{u,v} \notin \mathcal{V}\right)$
  - $\mathcal{E}' = \mathcal{E} \setminus \{\{(x, y) \mid (x = u) \vee (x = v)\}\}$
  $\cup \left\{\left\{\left(z_{u,v}, y\right) \mid (u, y) \in \mathcal{E} \vee (v, y) \in \mathcal{E}, y \neq u, y \neq v\right\}\right\}$

Properties

- $\sum_{v \in V} d(v) = 2m$
- $m \leq \binom{n}{2}$
- G is a tree $\Rightarrow m = n - 1$
- G is connected $\Rightarrow m \geq n - 1$
- G is acyclic $\Rightarrow m \leq n - 1$
- a connected graph is Eulerian $\Leftrightarrow$ every vertex has even degree
- in any (finite) graph the number of vertices of odd degree is even
- let $\mathcal{G} = (\mathcal{V}, \mathcal{E}), |\mathcal{V}| = n$. If $\mathcal{G}$ has a minimum cut of size t, then $|\mathcal{E}| \geq t \cdot \frac{n}{2}$

P vs NP

Definitions

- **P** is the set of decision problems that can be solved in polynomial time
- **NP** is the set of decision problems with the following property: if the answer is YES, then there is a proof of this fact that can be checked in polynomial time
- **CoNP** is the opposite of NP: if the answer is NO, then there is a proof of this fact that can be checked in polynomial time
- a computational problem is **NP-hard** if a polynomial time algorithm for it would imply a polynomial time algorithm for every problem in NP
- a problem is **NP-complete** if it is both NP-hard and in NP
- **reducing** problem A to problem B means describing an algorithm to solve A under the assumpion that an algorithm for B exists
- a problem A reduces in polynomial time to problem B $\left(A <_p B\right)$ if exists a polynomial time algorithm that transform an arbitrary input instance a of A into an input instance b of B such that:
  1. a is a YES instance of A $\Rightarrow$ b is a YES instance of B
  2. b is a YES instance of B $\Rightarrow$ a is a YES instance of A
- **NP-hardness** (formal definition): a problem is NP-hard if every problem in NP reduces in polynomial time to it

Properties

- $A <_p B$ and $B <_p C \Rightarrow A <_p C$

Optimization problems

- $\Pi : I \times S$ (I are the inputs and S the solutions)
- $c : S \to \mathbb{R}^+$ (cost of the solution)
- $\forall i \in I.\, S(i) = \{s \in S \mid i \Pi s\}$ (feasible solutions)
- $s^* \in S(i)$ and $c(s^*) = \min c(S(i))$ or $\max c(S(i))$ (optimal solution)
- Approximation: $s \in S(i)$ ok if $s \neq s^*$ but we need:
  1. guarantee on the quality of s
  2. guarantee on the complexity: polynomial time algorithm
- let Π an optimization problem, and let A_Π an algorithm for Π such that $\forall i \in I.\, A_\Pi(i) \in S(i)$. We say that A_Π has an **approximation factor** of $\rho(n)$ if $\forall i \in I s.t.\, |i| = n$ we have:

- $\frac{c(A_\Pi(i))}{c(s^*(i))} \leq \rho(n)$ (if we are considering a minimization problem)
- $\frac{c(s^*(i))}{c(A_\Pi(i))} \leq \rho(n)$ (if we are considering a maximization problem)

$c : S \to \mathbb{R}^+ \Rightarrow$ this two ratios are ≥ 1
- Goal: $\rho(n) = 1 + \varepsilon$ with ε as small as possible
- an **approximation scheme** for Π is an algorithm with two inputs $A_\Pi(i, \varepsilon)$ that $\forall \varepsilon$ is a $(1+\varepsilon)$-approximation
- an approximation scheme is polynomial (**PTAS**) if $A_\Pi(i, \varepsilon)$ is polynomial in $|i|$ $\forall \varepsilon$ fixed

Randomized algorithms

Definitions

- a **randomized algorithm** is an algorithm that may do a random choice
- the **space of probabilities** are the random choices made by the algorithm
- randomized algorithms that never fail are called **Las Vegas** algorithms
  - $\forall i \in I, A_R(i) = s \mid (i, s) \in \Pi$
  - s mat not be the same for all i
  - randomness comes into play in the analysis of the complexity
  - $\forall n, T(n)$ is a **random variable**, of which we usually study
    - $E[T(n)]$ (expected value) or
    - $\Pr(T(n) > c \cdot f(n)) \leq \frac{1}{n^k} \Rightarrow T(n) = O(f(n))$ "with high probability"
- randomized algorithms that may fail are called **Monte Carlo** algorithms
  - $i \in I$ it is possible that $A_R(i) = s \mid (i, s) \notin \Pi$
  - we study $\Pr((i, s) \notin \Pi)$ as a function of $|i| = n$
  - moreover, even $T(n)$ may be a random variable
  - decision problems can be divided into:
    - **one-sided** they may fail only on one answer
    - **two-sided** they may fail in both answers
- given $\Pi \subseteq I \times S$, an algorithm A_Π has complexity $T(n) = O(f(n))$ **with high probability** (w.h.p.) if $\exists$ constants $c, d > 0$ s.t. $\forall i \in I, |i| = n, \Pr(A_\Pi(i)$ terminates in $> c \cdot f(n)$ steps$) < \frac{1}{n^d}$
- given $\Pi \subseteq I \times S$, an algorithm A_Π is correct **w.h.p** if $\exists$ constant d s.t. $\forall i \in I, |i| = n, \Pr((i, A_\Pi(i)) \notin \Pi) \leq \frac{1}{n^d}$
- E_1, E_2 events are **independent** if $\Pr(E_1 \cap E_2) = \Pr(E_1) \cdot \Pr(E_2)$
- $\Pr(E_1) > 0$ then $\Pr(E_1 | E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_1)}$
- $\Pr(E_1 \cap E_2 \cap ... \cap E_k) = \Pr(E_1) \cdot \Pr(E_2 | E_1) \cdot \Pr(E_3 | E_1 \cap E_2) \cdot ... \cdot \Pr(E_k | E_1 \cap ... \cap E_{k-1})$

Properties

- **Markov's lemma**: let T be a non-negative, bounded ($\exists b \in \mathbb{N}$ $|\Pr(T > b) = 0$), integer random variable. Then $\forall t$ s.t. $0 \leq t \leq b$ we have $t \cdot \Pr(T \geq t) \leq E[T] \leq t + (b - t) \cdot \Pr(T \geq t)$

- **union bound lemma**: for any random events $E_1, E_2, ..., E_k$:
  $\Pr(E_1 \cup E_2 \cup ... \cup E_k) \leq \Pr(E_1) + \Pr(E_2) + ... + \Pr(E_k)$

Chernoff bound

Let $X_1, X_2, ..., X_n$ independent indicator random variables where $E[X_i] = p_i, 0 < p_i < 1$.

Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then $\forall \delta > 0$:

$$\Pr(X > (1+\delta) \cdot \mu) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu}$$

Variants of Chernoff bounds

1. $\Pr(X < (1-\delta) \cdot \mu) < e^{\frac{-\mu \cdot \delta^2}{2}}$ $(0 < \delta \leq 1)$
2. $\Pr(X > (1+\delta) \cdot \mu) < e^{\frac{-\mu \cdot \delta^2}{2}}$ $(0 < \delta \leq 2e - 1)$

Algorithms notes

DFS & BFS

- every vertex v has a field $L_V[v].\mathrm{ID} = \begin{cases} 1 \text{ if visisted} \\ 0 \text{ otherwise} \end{cases}$
- every edge e has a field $L_E[e].\mathrm{label} = \begin{cases} \text{null (initial value)} \\ \text{discovery edge} \\ \text{back edge} \end{cases}$
- problems that can be solved:
  - test if G is connected
  - find the connected components of G
  - find a spanning tree of G (if G is connected)
  - find a path between two vertices (if any)
  - find a cycle (if any)

Minimum spanning tree

- Both Prim and Kruskal, at each iteration A is a subset of edges of some MST
- Theorem: let $G = (V, E)$ be an undirected, connected and weighted graph. Let $A \subseteq E$ included in some MST of G, let $(S, V \setminus S)$ be a cut that repects A and let (u, v) a light edge for $(S, V \setminus S)$. Then (u, v) is safe for A (it can be added mantaining the invariant that A is included in some MST)
- If the weights are all distincts it exists exactly one MST

Floyd-Warshall

- call the vertices 1, 2, ..., n
- compute $\mathrm{dist}(u, v, k)$ = length of a shortest path from u to v that uses only vertices $\{1, 2, ..., k\}$ as internal (i.e. not u or v) and does not contain a directed cycle. (if no such path exists, define $\mathrm{dist}(u, v, k) = +\infty$)
- at each iteration: $A[u, v, k] = \min\{A[u, v, k-1], A[u, k, k-1] + A[k, v, k-1]\}$

Ford-Fulkerson

- the algorithm repeadetly finds an $s - t$ path P in G_f (e.g. using BFS) and uses P to increase the current flow. P is called augmenting path.

3-SAT

- a boolean formula is in conjunctive normal form (CNF) if it is a conjunction (AND) of several clauses, each of which is the disjunction (OR) of several literals, each of which is either a variable or its negation
- example: $(a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee e \vee f) \wedge (a \vee c \vee \neg e)$
- a 3-CNF formula ia a CNF formula with exactly 3 literals per clause

Hamiltonian Circuit

- given an undirected graph, an Hamiltonian Circuit is a cycle that traverses all the vertices only once

Maximum Independent Set

- given a graph $G = (V, E)$, an independent set in G is a subset $I \subseteq V$ with no edges between them
- compute an independent set of maximum size

Maximum Clique

- compute the largest complete subgraph in a given graph

Maximum Independent Set, Maximum Clique and Minimum Vertex Cover are equivalent

Approx Vertex Cover

- choose any edge
- add its endpoints to the solution
- remove the touchedd edges
- repeat

Travelling Salesman Problem (TSP)

- given a complete undirected graph and a function $w : E \to \mathbb{R}$, output a tour $T \subseteq E$ minimizing $\sum_{e \in T} w(e)$
- $w : E \to \mathbb{R}^+$ is without loss of generality because every tour has the same amount of edge, so we can add a large weight to each edge.
- theorem: for any function $\rho(n)$ that can be computed in polynomial time in n, there is no polynomial-time $\rho(n)$-approximation algorithm for TSP, unless P = NP (proved with a reduction from Hamiltonian Circuit)

Metric TSP

- a special case of TSP where the weight funtion w satisfies the triangle inequality: $\forall u, v, z \in V \,.\, w(u, v) \leq w(u, z) + w(z, v)$
- Metric TSP is NP-hard
- 2-approximation algorithm:

1. run Prim to obtain an MST T^*
2. return the vertices in the preorder of T^*

- 3/2-approximation
  1. run Prim to obtain an MST $T^* = (V, E^*)$
  2. let D be the set of vertices of T^* with odd degree. Compute a min-weight perfect matching M^* on the graph induced by D
  3. the graph $(V, E^* \cup M^*)$ is Eulerian, compute an Eulerian cycle on this graph
  4. return the cycle that visits all the vertices of G in the order of thei first appearence in the Eulerian cycle

Set Cover

- $I = (X, F)$
- $X =$ set of objects usually called "universe"
- $F \subseteq \{S \mid S \subseteq X\} =$ set of all subsets of X
- $\forall x \in X. \exists S \in F \mid x \in S$ (F covers X)
- optimization problem: find $F' \subseteq F$ s.t.
  1. F' covers X
  2. $|F'|$ is minimized
- Approximation:
  1. choose the subset that contains the largest number of uncovered elements
  2. remove from X the uncovered elements
  3. repeat

Karger's algorithm for Minimum Cut

- compute the cut of minimum size of a graph, in other words is the minimum number of edges that disconnects the graph
- we'll solve a more general problem: minimum cut on multi-graphs (moultiple edges between two vertices are allowed)
- $\mathcal{S} = \{\{\text{objects}\}\}$ is a multiset, $\forall$ objects $o \in \mathcal{S}. m(o) \in \mathbb{N} \setminus \{0\}$ (multiplicity of o in $\mathcal{S}$)
- Full Contraction:
  1. choose an edge at random
  2. "contract" the two vertices of that edge, removing all the edges incident both vertices
  3. repeat until only 2 vertices remain
  4. return the edges between them
- in order to obtain $\Pr(\text{Karger succeds}) > 1 - \frac{1}{n^d}$ we need to repeat "Full contraction" $k = \frac{dn^2 \ln n}{2}$ times

Algorithms complexities

Polynomial time algorithms

- **DFS & BFS**: $\Theta(n + m)$ in order to visit all the graph

- **Prim & Kruskal**: $O(m \cdot \log n)$
- **Prim without min-heap**: $O(m \cdot n)$
- **Kruskal without DSU**: $O(m \cdot n)$
- **DSU**:
  - init: $O(n)$
  - union: $O(\log n)$
  - find: $O(\log n)$
- **Dijkstra**: $O((m + n) \cdot \log n)$
- **Dijkstra without min-heap**: $O(m \cdot n)$
- **Bellman-Ford**: $O(m \cdot n)$ (in 2022 was published a near-linear version)
- **Floyd-Warshall**: $O(n^3)$
- **ford fulkerson**:
  - complexity: $O(m \cdot |f^*|)$ where f^* is a max flow (the flow value increases by ≥ 1 in each iteration and the cost of each iteration is m)
  - input size: $O(m \cdot \log U)$ where $U =$ max capacity
  - so $O(m \cdot |f^*|) = O(m \cdot n \cdot U)$ (pseudo-polynomial)
- **min-weight perfect matching** can be done in polynomial time
- **Karger**:
  - Full_contraction $= O(n^2) \Rightarrow$ Karger $= O(n^4 \cdot \log n)$ (version seen in class)
  - can be improved (Karger-Stein) to $O(n^2 \cdot \log^3 n)$
  - world record: $O(m \cdot \log n)$

NP-hard algorithms and approximations

- Cook-Levin Theorem: 3-SAT is NP-hard
- Hamiltonian Circuit is NP-hard
- Max Independent Set is NP-Hard (3Sat $<_p$ Max Ind. Set)
- Maximum clique is NP-hard (Max Ind. Set $<_p$ Max Clique)
- Minimum vertex cover is NP-hard
  - Max Ind. Set $<_p$ Min vertex cover
  - it exists a 2-approximation
- TSP is NP-hard (Ham $<_p$ TSP)
- Metric-TSP is NP-hard (TSP $<_p$ Metric-TSP)
- Set cover in its decision version $\langle (X, F), k \rangle$ is NP-hard
  - vertex cover $<_p$ set cover
  - it exists a $\lceil \log_2 n \rceil + 1$ approximation where $n = |X|$