Algorithm Theory, Tutorial 5

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

January 2019

Exercise 1, Pinning Polygons

You have two rectangular sheets of paper of equal dimensions. On each sheet an adversary has drawn straight lines that form a subdivision of each sheet into n polygons such that each polygon covers an equal area. The subdivision is different for each sheet. You also receive n pins.

(a) You put *two* of the sheets directly on top of each other. Prove that your pins suffice to pierce *all* polygons on both sheets (no folding or other funny business, polygons need to be pinned through their *interior*).

(b) Show that this is not possible if we drop the condition that each polygon covers an equal area on each sheet (however, the area of each polygon is bigger than zero).

Solution 1a

- Core idea: This is a matching problem, use hall's theorem
- Define bipartite graph $(L \cup R)$ with $L :=$ Polygons in upper sheet and $R :=$ polygons in lower sheet and connecting edges if two polygons can be connected with a pin.
- Perfect Matching in this graph means that n pins suffice.
- each polygon has the same size s.
- Let $A \subset L$
- A covers an area of $|A| \cdot s$ in the upper sheet.
- An area of this size covers at least $|A$ polygons in the lower sheet.
- $\Rightarrow |N(A)| \geq |A|$
- $\Rightarrow$ Perfect matching exists, n pins suffice.
- I don't think we need "polygons created by straight lines" to make this work.

Solution 1b

Construct counterexample that violates Hall's theorem

Exercise 2, Doomsday

Horrible news on Krypton: The planet will be struck by a meteorite within the next m minutes and the planet, all of its n cities and all k Kryptonions inhabiting them will be destroyed. Fortunately the Kryptonian government provided interstellar escape pods for such an emergency.

In city i, where k_i Kryptonians live ($\sum_{i=1}^{n} k_i = k$), there are p_i such pods available. Each pod can carry one person. Kryptonians can either *instantly* use one of the escape pods in the city where they live, or travel to another city and use an escape pod there. It takes d_{ij} minutes to travel from city i to city j. A pod must be reached before the meteorite destroys Krypton.

Due to flight safety concerns the total number of pods that can launch from certain subsets of cities is restricted. I.e., there are subsets $S_i \subseteq [1..n]$ with $i \in [1..\ell], \ell \leq n$ and parameters r_i. Each city j is subject to exactly one flight restriction zone (that is $j \in S_i$). All cities that are part of S_i can not launch more than r_i pods taken together.

Doomsday, Continuation

(a) You need to determine the maximum number of survivors. Describe how this problem can be reduced to a maximum flow problem.

(b) Making no assumptions about k, ℓ, n, m other than the ones given in the exercise, how long does it take in the worst case to solve the maximum flow problem using the Ford-Fulkerson algorithm?

Solution 2a

- Flow problem with multiple layers
- First layer: Cities in which the kryptonians currently are.
- ....

Solution 2a, Drawing a picture together

Solution 2a, Formalized

We define a flow network as follows. First we create a set of nodes $C_1, \ldots, C_n$ for the cities and connect the source s with each city, whereas each of these edges has capacity $k_1, \ldots, k_n$, corresponding to the number of inhabitants of the respective city.
Now we create another set of nodes $P_1^1, \ldots, P_n^1, P_1^2, \ldots, P_n^2$ for the escape pods. We establish an edge from P_j^1 to P_j^2 for each $1 \leq j \leq n$, and assign it capacity p_j (representing the number of pods available in city j). Then we create an edge from C_i to P_j^1, iff $d_{i,j} \leq m$ (i.e. if the time suffices to travel from city i to a pod in city j). These edges are not restricted in capacity, so we assign capacity k to each.
Finally we have to take care of the flight restriction zones. We create nodes $R_1, \ldots, R_\ell$ and create an edge from P_j^2 to R_i, iff $j \in S_i$. We create an edge from each R_i to the sink and assign these edges the capacity r_i (to restrict the number of pods that can launch from the respective cities).

Solution 2b

- n cities, k inhabitants
- Edges from source to cities: $O(n)$
- Edges from cities to Pods:
- Edges from pods to restrictions:
- Edges from Restrictions to sink:
- Edges total:
- Maximum flow is at most:
- Total runtime according to FF:

Sinkless orientation, 3a

- Let $B = (U \cup V, E)$ be a bipartite graph and assume that for every $A \subseteq U$, we have $|N(A)| \geq |A|$. Show that this implies that there exists a matching of size $|U|$ in B (i.e., a matching that matches every node in U).
- What is the difference to hall's theorem?

Solution 3a

- Our claim is similar to Hall's Theorem, we will also use a similar approach for the proof.
- We will use a proof by contradiction, this means we actually show:

$$B \text{ has } \textbf{no} \text{ matching of size } |U| \Rightarrow \exists A \subseteq U : |N(A)| < |A|$$

- Construct Flow network similar to lecture:
  - New nodes s and t (source and sink)
  - Edges from s to each node in U with capacity 1
  - Edges from each node in V to t with capacity 1
  - All edges from original graph become directed from U to V.

Solution 3a

- Maximum flow must be less than $|U|$ (otherwise a matching of the same size exists, see lecture).
- Thus any min cut $C = (A', B')$ has size $|C| < |U|$.
- We assume wlog that $s \in A'$ (otherwise switch the names)
- Some observations:
  - A' is not empty and does not contain all nodes (otherwise no valid cut)
  - $A' \neq \{s\}$ (the cut would have size $|U|$)
  - $A' \neq \{s\} \cup U \cup V$ (the cut would have size $|U|$)
  - $A := A' \cup U$ is not empty.

Solution 3a

- Idea: Divide the min-cut edges into three groups:
  - Connecting s to B'
  - Connecting nodes in U to B'
  - Connecting nodes in V to B'.
- Let x be the number of edges from s to B'.
- We know that $x = |U \setminus A| = |U| - |A|$ $(*)$.
- Let z be the number of edges from A to B'.
- Let y be the number of edges from $A' \setminus (A \cup s) = A' \cap V$ to B'.
- Then $y = |A' \cap V|$ since by construction each node in V has exactly one edge to the sink t.
- Our total min-cut has size

$$|C| = x + y + z < |U|$$

Solution 3a

- Reminder: $A = A' \cap U$ (nodes in U that are in source side of min-cut)
- Consider the neighborhood $N(A)$ (wrt to original graph without s, t).
- we know that $|N(A)| \leq y + z$ $(***)$
- $N(A)$ can not contain more nodes then the edges going from A to B' (which is z), plus the nodes in $A' \cap V$ (which is y).
- Together:

$$|N(A)| \leq y + z < |U| - x = |A|$$

Exercise 3b

- Let $G = (V, E)$ be a graph with minimum degree at least two (i.e., each node has at least two incident edges). Use the prior statement to show that there is a way to orient the edges of G such that each node of G has at least one out-going edge (this is known as a sinkless orientation).
  Hint: Sinkless orientation can be seen as matching nodes and edges.

Solution 3b

- Use the hint, consider matching between nodes and edges.
- If a node v and an edge e are matched this means that the e will be directed away from v in our sinkless orientation.
- Formalize bipartite Graph $(V \cup E)$ where the original nodes AND edges become nodes.
- v and e are connected in bipartite graph iff EDGE e was connected to node v in original graph (example)

Solution 3b

- $v \in V$ has degree ≥ 2 in bipartite graph (because of degree in original graph)
- $e \in E$ has degree **exactly** 2 (same)
- $A \subset V$ has at least 2 $|A$ outgoing edges, these hit at least $|A|$ different elements of E.
- $N(A) \geq |A|$ for all $A \subset V \Rightarrow$ Matching exists (Result from 3a)

Exercise 3c

- Let us now assume that the graph G has minimum degree 4. Show that there exists an orientation in which each node has at least one in-coming and at least one out-going edge.

Solution 3c

- Constructe the bipartite graph as in 3b
- Split each node $v \in V$ into two nodes v_1 and v_2
- v_2 and v_2 each get half of the edges of v (at least two)
- We now compute the matching which still exists.
- Each $v \in V$ is now matched to two edges (one from v_1 and one from v_2) which it can arbitrarily orient.

Ex 4 Triangles in random graphs

Given a fixed vertex set $V = \{v_1, v_2, \ldots, v_n\}$ with n being an even number. Then the following (randomized) process defines the (undirected) random graph $G_p = (V, E_p)$.

For each vertex pair $\{v_i, v_j\}, i \neq j$ we independently decide with probability p whether the edge defined by this pair is part of the graph, i.e., whether $\{v_i, v_j\}$ is an element of the edge set E_p.

Furthermore we say that a subset $T = \{v_i, v_j, v_k\}$ of V of size 3 is a triangle of a graph, if all three edges $\{v_i, v_j\}$, $\{v_i, v_k\}$, $\{v_j, v_k\}$ are in the edge set of the graph.

(a) Let Z be the random variable that counts the number of edges in G_p. What is the distribution of Z? What is the probability that Z has value k, for some k?

Solution 4a

- Z is the sum of independent "coin tosses" (random variables that either take 0 or 1).
- This is the definition of a binomial distribution
- number of single variables : $\binom{n}{2}$ (each pair of nodes is a possible edge)
- Plug in formula for Probability:
  $Z \sim Bin\left(\binom{n}{2}, p\right)$.

$$\Pr(Z = k) = \binom{\binom{n}{2}}{k} p^k (1-p)^{\binom{n}{2}-k}$$

4b

- Calculate m_T, the number of all triangles that could *possibly* occur in G_p.
- Each node can appear only once per triangle
- We do not care about the order of nodes in a triangle
- Thus we have $m_t = \binom{n}{3}$ possible triangles. (Basic combinatorics)

4c

- Let X denote the number of triangles in G_p. Calculate $\mathbb{E}[X]$.
- For each triangle the probability that this triangle is part of the graph is p^3 (we have to draw all three edges.)
- Because we have $\binom{n}{3}$ triangles and because of the linearity of expectation we directly get

$$\mathbb{E}[X] = \binom{n}{3} \cdot p^3$$

Exercise 4d

The generation of the random graphs is now changed as follows. Before edges are determined each vertex is colored either red or green; we let K be the random variable that counts the number of red vertices. Between two red vertices there is an edge with probability p_{rr}, between two green vertices with probability p_{gg} and between vertices of different color with probability p_{rg} (edges are still picked independently).

(d) Assume first that with probability $\frac{1}{7}$ all vertices are red, with probability $\frac{2}{7}$ all vertices are green and with probability $\frac{4}{7}$ each vertex independently gets color red or green with probability $1/2$ each. Also $p_{rr} = 1$, $p_{rg} = \frac{1}{\sqrt{3}}$ and $p_{gg} = 0$. Calculate $\mathbb{E}[\mathbb{X}]$ under these conditions!

Solution 4d

Let us define the following events:
$\mathcal{A}$: is the event when all the vertices are red.
$\mathcal{B}$: is the event when all the vertices are green.
$\mathcal{C}$: is the event when each vertex gets color red or green with probability $1/2$.

$$\Pr(\mathcal{A}) = \frac{1}{7}, \quad \Pr(\mathcal{B}) = \frac{2}{7}, \quad \Pr(\mathcal{C}) = \frac{4}{7}$$

$$E[X] = E\left[\sum_{i=1}^{\binom{n}{3}} X_i\right] = \sum_{i=1}^{\binom{n}{3}} E[X_i] \qquad \text{[from the linearity of expectation]}$$

$$= \sum_{i=1}^{\binom{n}{3}} \Pr(X_i = 1)$$

$$= \sum_{i=1}^{\binom{n}{3}} \Big(\Pr(X_i = 1|\mathcal{A}) \Pr(\mathcal{A}) + \Pr(X_i = 1|\mathcal{B}) \Pr(\mathcal{B}) +$$

$$\Pr(X_i = 1|\mathcal{C}) \Pr(\mathcal{C}) \Big) \quad \text{[law of total probability]}$$

$$= \sum_{i=1}^{\binom{n}{3}} \Big(p_{rr}^3 \Pr(\mathcal{A}) + p_{gg}^3 \Pr(\mathcal{B}) + \Pr(X_i = 1|\mathcal{C}) \Pr(\mathcal{C}) \Big)$$

$$\overset{**}{=}$$

$$\sum_{i=1}^{\binom{n}{3}} \left(1 \cdot \left(\tfrac{1}{7}\right) + 0 \cdot \left(\tfrac{2}{7}\right) + \left(\tfrac{1}{8} \cdot 1^3 + \tfrac{1}{8} \cdot 0 + \tfrac{3}{8} \cdot 1 \cdot \tfrac{1}{\sqrt{3}} \cdot \tfrac{1}{\sqrt{3}} + \tfrac{3}{8} \cdot 0 \cdot \tfrac{1}{\sqrt{3}} \cdot \tfrac{1}{\sqrt{3}}\right) \cdot \tfrac{4}{7}\right)$$

$$= \binom{n}{3} \cdot \left(\tfrac{1}{7} + \tfrac{2}{8} \cdot \tfrac{4}{7}\right) = \tfrac{2}{7} \cdot \binom{n}{3}$$

** For calculating $\Pr(X_i = 1|\mathcal{C})$ we apply the law of total probability on the color of the three nodes in the i^{th} triple of vertices. Note that the probability of having the three vertices all red or all green is $1/8$ each, having two red vertices and one green vertex is $3/8$, and one red vertex and two green vertices is $3/8$.