Exercise Set XI, Advanced Algorithms 2022

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

These problems are taken from various sources at EPFL and on the Internet, too numerous to cite individually. **The problems are not ordered with respect to difficulty.**

1 Let M be the normalized adjacency matrix of a d-regular undirected graph $G = (V, E)$. In class, we proved that the maximum eigenvalue equals 1.

Show that the maximum *absolute* value of an eigenvalue is at most 1. That is, for any eigenvalue λ of M, we have $|\lambda| \leq 1$.

Solution: Let x be any eigenvector of M and let λ be the corresponding eigenvalue. Let $y = \lambda x = Mx$, and let $i \in V$ be such that $|x(i)|$ is maximized. Then

$$|(\lambda x)_i| = |y(i)| = \left| \sum_{(i,j)\in E} \frac{x(j)}{d} \right| \leq \sum_{(i,j)\in E} \left| \frac{x(j)}{d} \right| \leq \sum_{(i,j)\in E} \left| \frac{x(i)}{d} \right| = |x(i)|,$$

which gives $|\lambda| \leq 1$.

2 Let M be the normalized adjacency matrix of a d-regular undirected graph $G = (V, E)$ that is connected. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of M. Show that $\lambda_n = -1$ if and only if G is bipartite.

(Hint: to show $\lambda_n = -1$, we only need to find a vector x such that $Mx = -x$.)

Solution: First, let's assume that G is bipartite, $V = V_1 \cup V_2$. We consider the vector x defined by:

$$x_i = \begin{cases} 1 & \text{if } i \in V_1 \\ -1 & \text{if } i \in V_2 \end{cases}$$

Since every vertex in V_1 has its neighbors in V_2, and conversely, we have $Mx = -x$. Therefore, -1 is an eigenvalue for M and it has to be the smallest, since the absolute value of every eigenvalue of M is bounded by 1 (see last exercise). Hence $\lambda_n = -1$.

We now assume $\lambda_n = -1$ and we consider an eigenvector x associated with λ_n. Let $x_i = D$ be the largest component of x in absolute value. Since $(Mx)_i = -x_i$ is the average of the value of the d neighbours of i, it means that those neighbors have to be associated with the value $-D$. The same goes for the neighbors of the neighbors of i which have to be associated with the value D. Since the graph is connected, we can extend the reasoning to every vertex.

We just proved that each component of x is either D or $-D$. Let $V_1 = \{i : x_i = D\}$ and $V_2 = \{i : x_i = -D\}$. We also proved that every neighbor of a vertex in V_1 has to be in V_2, and conversely. Therefore, G is bipartite with $V = V_1 \cup V_2$.

3 In spectral graph theory, a popular matrix is the (normalized) Laplacian matrix. Its definition (for d-regular graphs) is as follows. Let M be the normalized adjacency matrix of a d-regular undirected graph $G = (V, E)$. The normalized Laplacian matrix is $L = I - M$.

3a Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of M. Show that $(1-\lambda_1) \leq (1-\lambda_2) \leq \ldots \leq (1-\lambda_n)$ are the eigenvalues of L.

Solution: Let v_i be the eigenvector of M, with eigenvalue λ_i, hence $Mv_i = \lambda_i v_i$. Also note that $Iv_i = v_i$. Thus we have

$$Lv_i = (I - M)v_i = Iv_i - Mv_i = v_i = \lambda_i v_i = (1 - \lambda_i)v_i.$$

Thus, v_i is the eigenvector of M, with the eigenvalue $1 - \lambda_i$

3b One reason for the popularity of the normalized Laplacian matrix is because its quadratic form $x^\top Lx$ highlights the connection between cuts and eigenvectors. Indeed, verify the following identity

$$x^\top Lx = \frac{1}{d} \sum_{\{i,j\} \in E} (x(i) - x(j))^2 \ .$$

Notice that if $x \in \{0,1\}^n$, then the above identity says that $x^\top Lx$ equals the number of edges cut by the set $S = \{i : x(i) = 1\}$ normalized by $\frac{1}{d}$.

Solution: Recall that $M = \frac{1}{d}A$ with A being the d adjacency matrix of G. Let $x \in \mathbb{R}^n$ be a function on the vertices of $G = (V, E)$. Then

$$\begin{aligned}
x^\top Lx &= x^\top (I - M)x = x^\top \left(I - \frac{1}{d}A\right) x = x^\top x - \frac{1}{d}(x^\top Ax) \\
&= \sum_{u \in V} x(u)^2 - \frac{1}{d}\sum_{u \in V} x(u) \sum_{v:\{u,v\} \in E} x(v) \\
&= \frac{1}{d}\left(\sum_{u \in V} dx(u)^2 - \sum_{u \in V}\sum_{v:\{u,v\} \in E} x(u) \cdot x(v)\right) \\
&= \frac{1}{d}\sum_{u \in V}\sum_{v:\{u,v\} \in E} \left(x(u)^2 - x(u) \cdot x(v)\right) \\
&= \frac{1}{d}\sum_{\{u,v\} \in E} (x(u)^2 + x(v)^2 - 2x(u) \cdot x(v)) \\
&= \frac{1}{d}\sum_{\{u,v\} \in E} (x(u) - x(v))^2 \ .
\end{aligned}$$

4 Let $G = (V, E)$ be a d-regular undirected graph $G = (V, E)$. In this problem we shall analyze the *lazy* random walk on G:

- With probability 1/2: we stay at the current vertex
- With remaining probability 1/2: we go to a random neighbor (out of d possibilities).

4a Show that the smallest eigenvalue of the matrix corresponding to a lazy random walk is at least 0.

Solution:

- Let p be the current distribution over vertices and let q be the distribution after taking one random step.
- In a lazy random walk:
  - with probability 1/2: we don't move so the distribution doesn't change: $q = Ip$.
  - with probability 1/2: we take a random step as in the normal random walk: $q = Mp$, where M is the normalized adjacency matrix.
- Hence the matrix corresponding to random walk is $\frac{1}{2} \cdot I + \frac{1}{2} \cdot M$. Let λ be an arbitrary eigenvalue and let v be the corresponding eigenvector of this matrix. Then we have

  $$(\tfrac{1}{2} \cdot I + \tfrac{1}{2} \cdot M)v = \lambda \cdot v \Rightarrow M \cdot v = (2\lambda - 1) \cdot v,$$

  and thus, $2\lambda - 1$ is an eigenvalue of M. Since all eigenvalues of M are at least -1, we have that $2\lambda - 1 \geq -1 \Rightarrow \lambda \geq 0$.

Some questions related to any subject of the course

5 Given a graph $G = (V, E)$ and an integer k, design a randomized algorithm that returns a coloring $c : V \rightarrow \{1, \ldots, k\}$ such that in expectation at least $\left(1 - \frac{1}{k}\right)$-fraction of the edges are *correctly colored*. An edge $e = \{u, v\}$ is correctly colored if it is not monochromatic, i.e., $c(u) \neq c(v)$.

Solution: The randomized algorithm is simple: give each vertex one of the k colors independently and uniformly at random. We say that an edge $\{u, v\} \in E$ is *satisfied* if u and v have a different color. To compute the expected number of satisfied edges, we introduce for each $e \in E$ a random variable X_e which is set to one if e is satisfied and zero else. First, we compute the probability that $X_{u,v} = 1$:

$$\Pr(X_{u,v} = 1) = 1 - \Pr(u \text{ and } v \text{ have the same color}) = 1 - \frac{1}{k}$$

We can now simply compute the expected number of satisfied edge using linearity of expectation:

$$\mathbb{E}[\text{number of satisfied edges}] = \sum_{e \in E} \mathbb{E}[X_e] = |E| \cdot \left(1 - \frac{1}{k}\right)$$

$\square$

6 Consider the matching problem in general graphs. Suppose that you are given a black-box polynomial-time algorithm $\mathcal{A}$ which, given a graph, returns TRUE if the graph has a perfect matching and NO otherwise[1] Explain how to use $\mathcal{A}$ in order to, in polynomial time,

a find a perfect matching in a graph,

b find a maximum cardinality matching in a graph.

[1]For example, the algorithm obtained by calculating the determinant of the Tutte matrix that we saw in class.

Solution: a) The following algorithm might not be the most efficient but its correctness is easy to establish and it is polynomial:

```
**Find perfect matching given $G = (V, E)$ that has a perfect matching**

  1. While $|E(G)| > |V|/2$:
      (a) find $e \in E(G)$ such that $\mathcal{A}(G - e) = \texttt{TRUE}$
      (b) $G \leftarrow G - e$
  2. Return $E(G)$
```

Notice that since a perfect matching touches all the vertices exactly once, it must have $|V|/2$ edges. Thus, if $|E(G)| > |V|/2$ then there must exists some removable $e \in E$. Also, notice that G having a perfect matching is a loop invariant guaranteed by the oracle. Thus our algorithm returns a perfect matching and runs in time polynomial (provided that $\mathcal{A}$ runs in time polynomial too).

b) We're going to allow for gradually worse matching in G by adding more and more vertices that are connected to everything in G and thus can "remove" some vertex from a matching. For sake of simplicity of notation let us assume that n the number of vertices is even.

```
**Find a maximum cardinality matching in $G = (V, E)$**

  1. For $i = 0, 2, 4, \ldots, n$:
      (a) Let $G_i$ be $G$ where we have added $i$ vertices and connected those to every $v \in V$
      (b) If $\mathcal{A}(G_i) = \texttt{TRUE}$ then let $M$ be a perfect matching of $G_i$ given by the previous
          algorithm. Return $E \cap M$.
```

The above algorithm works with the following simple observation: For each $i = 0, 2, 4, \ldots, n$, G has a matching of size $\frac{n-i}{2}$ if and only if G_i has a perfect matching. If n is odd, then simply iterate for $i = 1, 3, \ldots, n$ (it makes sense to start at $i = 1$, there is no perfect matching for an odd number of vertices).