

You have 90 minutes to answer four of these questions.  
**Write your answers in the separate answer booklet.**  
 You may take the question sheet with you when you leave.

**Chernoff Bounds:** If  $X$  is the sum of independent indicator variables and  $\mu = E[X]$ , then the following inequalities hold for any  $\delta > 0$ :

$$\Pr[X < (1 - \delta)\mu] < \left(\frac{e^{-\delta}}{(1 - \delta)^{1-\delta}}\right)^\mu \quad \Pr[X > (1 + \delta)\mu] < \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^\mu$$

1. Prove that the following problems are NP-complete:
  - (a) [5 pts] Given a graph  $G$  and a positive integer  $k$ , decide whether there is a spanning tree of  $G$  that has *at most*  $k$  leaves.
  - (b) [5 pts] Given a graph  $G$  and a positive integer  $k$ , decide whether there is a spanning tree of  $G$  that has diameter *at least*  $k$ . (The diameter of a tree is the maximum length of a path between two leaves.)
  
2. This problem asks you to analyze what happens when you throw balls randomly into bins. There are  $n$  bins. Each time you throw a ball, it lands in a random bin. Each bin is equally likely, and the results of different throws are independent.
  - (a) [3 pts] Suppose  $k$  of the  $n$  bins are initially empty, and you stop just after a ball lands in an empty bin. What is the expected number of throws? Prove your answer is correct.
  - (b) [4 pts] Now suppose initially every bin is empty, and you stop as soon as *every* bin is occupied. What is the expected number of throws? Prove your answer is correct.
  - (c) [3 pts] How many balls must you throw to ensure that *with high probability* every bin is non-empty? Prove your answer is correct.
  
3. This problem asks you to compute some probabilities associated with random walks. The setting is a row of  $n$  squares, one of which contains a token. On each turn, the token moves either one step to the right or one step to the left, each with probability  $1/2$ . The game ends when the token falls off the board, by moving either left from the leftmost square or right from the rightmost square.
  - (a) [5 pts] Suppose the token starts on the leftmost square. Prove that the probability that the token falls off the right end of the board is exactly  $1/(n + 1)$ .
  - (b) [5 pts] Suppose the token starts on the  $k$ th square from the left. What is the probability that the token falls off the right end off the board? Prove your answer is correct. [Hint: This should imply part (a).] A correct answer without proof is worth 2 points.

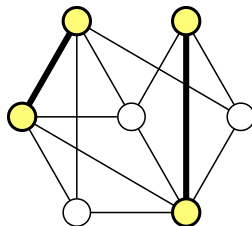


Problem 3(b) with  $k = 5$  and  $n = 12$

4. Consider the following modification of the 2-approximation algorithm for minimum vertex cover that we saw in lecture. The only real change is that we compute a set of edges instead of a set of vertices.

<p>APPROXMINMAXMATCHING(<math>G</math>):</p> <p><math>M \leftarrow \emptyset</math></p> <p>while <math>G</math> has at least one edge</p> <p>    <math>(u, v) \leftarrow</math> any edge in <math>G</math></p> <p>    <math>G \leftarrow G \setminus \{u, v\}</math></p> <p>    <math>M \leftarrow M \cup \{(u, v)\}</math></p> <p>return <math>M</math></p>
--

- (a) [2 pts] Prove that the output set  $M$  is a *matching*—no pair of edges in  $M$  have a common vertex.
- (b) [2 pts] Prove that  $M$  is a *maximal* matching— $M$  is not a proper subgraph of any other matching in  $G$ .
- (c) [6 pts] Prove that  $M$  contains at most twice as many edges as the *smallest* maximal matching in  $G$ .



The smallest maximal matching in a graph.

5. A *cut* in a graph  $G$  is a partition of the vertices of  $G$  into two classes: black and white, winners and losers, Democrats and Republicans, or whatever. The *size* of a cut is the number of edges that have one endpoint in each class. In lecture we saw a randomized algorithm that computes a cut of minimum size, with high probability, in polynomial time. The corresponding *maximum* cut problem is NP-hard.

Prove that the following randomized algorithm computes a cut whose *expected* size is at least half the size of the largest cut in  $G$ . In other words, show that this is an expected 2-approximation algorithm.

<p>DUMBMAXCUT(<math>G</math>):</p> <p>for each vertex <math>v</math> in <math>G</math></p> <p>    with probability <math>1/2</math></p> <p>        color <math>v</math> white</p> <p>    else</p> <p>        color <math>v</math> black</p>
---