

UNIVERSITY OF BRISTOL

January 2021 Examination Period

Department of Computer Science

2nd Year Examination for the Degrees of  
Bachelor in Computer Science  
Master of Engineering in Computer Science  
Master of Science in Computer Science

COMS20010  
Algorithms II

**TIME ALLOWED:**

**2 Hours**

**plus 30 minutes to allow for collation and uploading of answers.**

This paper contains **twenty-two** questions.

**All** questions will be marked.

If you attempt a question and do not wish it to be marked, delete it clearly.

The maximum for this paper is **150 marks**.

Other Instructions

1. You may access any materials available on the COMS20010 unit page and any materials which you have created yourself, but no others.
2. You may use a calculator if you wish.

## Section 1 — Short-answer questions (75 marks)

You do not need to justify your answers for any of the questions in this section, and you will not receive partial credit for showing your reasoning. Just write your answers down in the shortest form possible, e.g. “A” for multiple-choice questions, “True” for true/false questions, or “23” for numerical questions. If you do display working, circle or otherwise indicate your final answer, as if it cannot be identified then the question will not be marked.

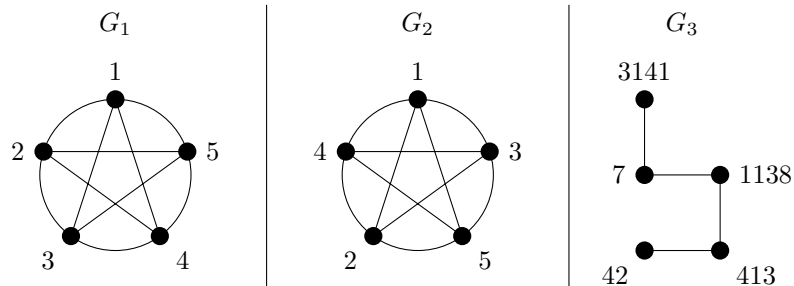
### Question 1 (5 marks)

(Short question.) For **each** of the following statements, identify whether it is true or false.

- (a)  $n \in O(n^2)$ . (1 mark)
- (b)  $n \in \omega(\log^2 n)$ . (1 mark)
- (c)  $n^2 + 50n - 12 \in O(\frac{1}{2}n^2 - n + 100)$ . (1 mark)
- (d)  $5^n \in \Theta(6^n)$ . (1 mark)
- (e)  $mn^2 + m^2n \in O(m^{100}n)$ . (1 mark)

### Question 2 (5 marks)

(Short question.) Consider the three graphs  $G_1$ ,  $G_2$  and  $G_3$  shown below.



For **each** of the following statements, identify whether it is true or false.

- (a)  $G_1$  is equal to  $G_2$ . (1 mark)
- (b)  $G_2$  is isomorphic to  $G_2$ . (1 mark)
- (c)  $G_1$  contains a subgraph isomorphic to  $G_3$ . (1 mark)
- (d)  $G_1$  contains an induced subgraph isomorphic to  $G_3$ . (1 mark)
- (e)  $G_1$  contains an Euler walk from some vertex back to itself. (1 mark)

### Question 3 (5 marks)

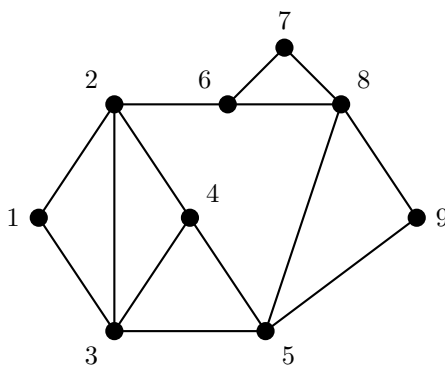
(Short question.) You are working on a recently-established competitor to Google Maps, and you have been asked to estimate storage requirements for adding the small landlocked nation of Tropicco to the service. Tropicco’s road network will be stored internally as a graph: Each junction and dead end is a vertex, and the roads joining them are edges. After doing some research, you discover that Tropicco’s road network has 100 dead ends,

350 3-way junctions, 50 4-way crossroads, and one terrifying central roundabout with 20 separate exits (which you should consider as a 20-way junction). How many edges will the resulting graph have? Choose **one** of the following options.

- A. 645.
- B. 685.
- C. 1290.
- D. 1370.
- E. None of the above.

**Question 4** (5 marks)

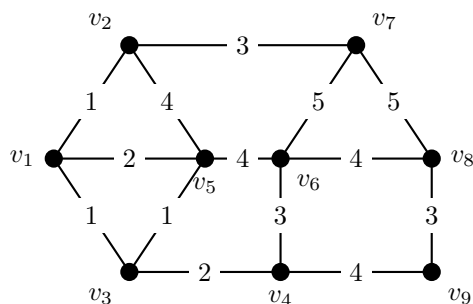
(Short question.) Consider a depth-first search in the following graph starting from vertex 1.



In the implementation of depth-first search given in lectures, we say vertex  $i$  is *explored* when `explored[i]` is set to 1. Which vertex will be explored sixth? Assume that whenever the search has a choice of two or more vertices to visit next, it picks the vertex with the lowest number first.

**Question 5** (5 marks)

(Short question.) Consider the following edge-weighted graph.

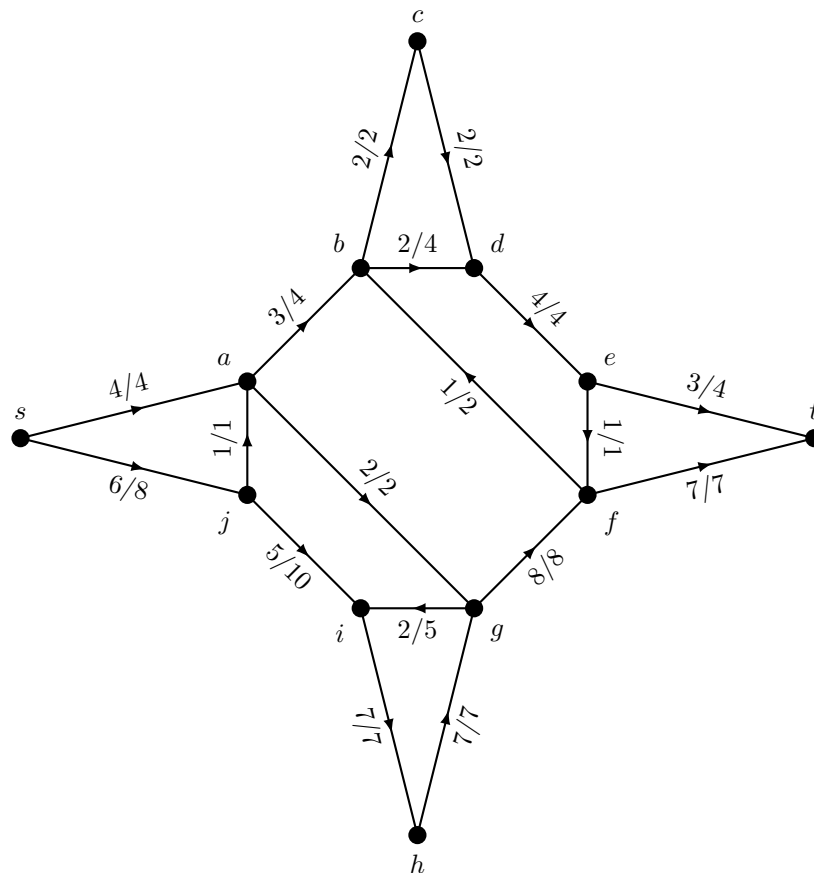


Which of the following edges will **always** be selected as part of a minimum spanning tree by Prim's algorithm starting from  $v_1$ , no matter how it breaks ties? Choose **one** of the following options.

- A.  $\{v_3, v_5\}$ .
- B.  $\{v_6, v_8\}$ .
- C.  $\{v_8, v_9\}$ .
- D. More than one of these edges will always be selected.
- E. None of these edges will always be selected.

**Question 6** (5 marks)

(Short question.) Consider the following vertex flow network with source  $s$ , sink  $t$  and flow  $f$ .



- (a) What is the value of  $f$ ? (1 mark)
- (b) What is  $f^-(a)$ ? (1 mark)
- (c) Is  $(\{s, b, e\}, \{a, c, d, f, g, h, i, j, t\})$  a valid cut? (1 mark)
- (d) Give an augmenting path for  $f$ . (2 marks)

**Question 7** (5 marks)

(Short question.) Which of the following CNF formulae are satisfiable?

- (a)  $x \vee \neg x$ . (1 mark)

(cont.)

(b)  $\neg a \wedge (a \vee \neg c) \wedge (a \vee c)$ . (1 mark)

(c)  $(x \vee y) \wedge (x \vee \neg y)$ . (1 mark)

(d)  $(\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (a \vee b \vee c)$ . (1 mark)

(e)  $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge \cdots \wedge (x_{99} \vee \neg x_{100}) \wedge (\neg x_{99} \vee x_{100})$ . (1 mark)

### Question 8 (5 marks)

(Short question.)

**Context:** You are attempting to steal a valuable work of art from the Tate Modern. By using the ventilation ducts, you believe you can steal any artwork not directly in view of a camera. With great difficulty, you have determined the number of cameras in the building and a limited set of points where they could be mounted. You also know, for each mounting point, which artworks a camera mounted there would protect, and you have recorded this information in the form of a bipartite graph. Unfortunately, you haven't been able to narrow down the cameras' locations exactly. You don't want to go to the trouble of breaking in and then leave empty-handed, so you would like to know: Is it true that *however* the cameras are mounted, you will be able to steal at least one artwork?

**Problem:** The ARTTHIEF problem is defined as follows. An instance of the problem consists of an integer  $k$ , a set  $A$  of artworks, a set  $P$  of mounting points, and a bipartite graph  $G$  with vertex classes  $A$  and  $P$ . The answer is **Yes** if for all sets of camera locations  $C \subseteq P$  with  $|C| = k$ , there exists  $a \in A$  which is not adjacent to any  $p \in P$ . Otherwise, the answer is **No**. Which of the following statements has a short and simple proof?

- (a) ARTTHIEF is in NP.
- (b) ARTTHIEF is not in NP.
- (c) ARTTHIEF is in Co-NP.
- (d) ARTTHIEF is not in Co-NP.
- (e) More than one of the above, or none of the above.

### Question 9 (5 marks)

(Short question.) Consider the “unoptimised” union-find data structure presented in lectures, in which a sequence of  $n$  operations has a worst-case running time of  $\Theta(n \log n)$  rather than  $\Theta(n\alpha(n))$ . Let  $G$  be the graph of such a data structure initialised with the following commands:

```
MakeUnionFind([10]);
Union(3, 4);
Union(3, 5);
Union(3, 6);
Union(1, 2);
Union(1, 4);
Union(7, 1).
```

(cont.)

- (a) How many components does  $G$  have? (2 marks)
- (b) What is the maximum depth of any component of  $G$ ? (Remember that depth is the greatest number of **edges** from the root to any leaf.) (3 marks)

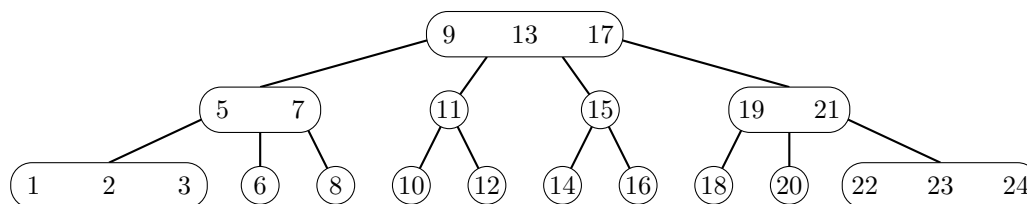
**Question 10** (5 marks)

(Short question.) You are implementing rudimentary pathfinding in a video game, trying to find the shortest paths from a large number of enemies to the player in a maze. Which of the following algorithms would be the most efficient choice for this situation?

- (a) Breadth-first search.
- (b) Depth-first search.
- (c) Dijkstra's algorithm.
- (d) The Bellman-Ford algorithm.
- (e) None of the above algorithms would work.

**Question 11** (5 marks)

(Medium question.) Let  $T$  be the 2-3-4 tree below.

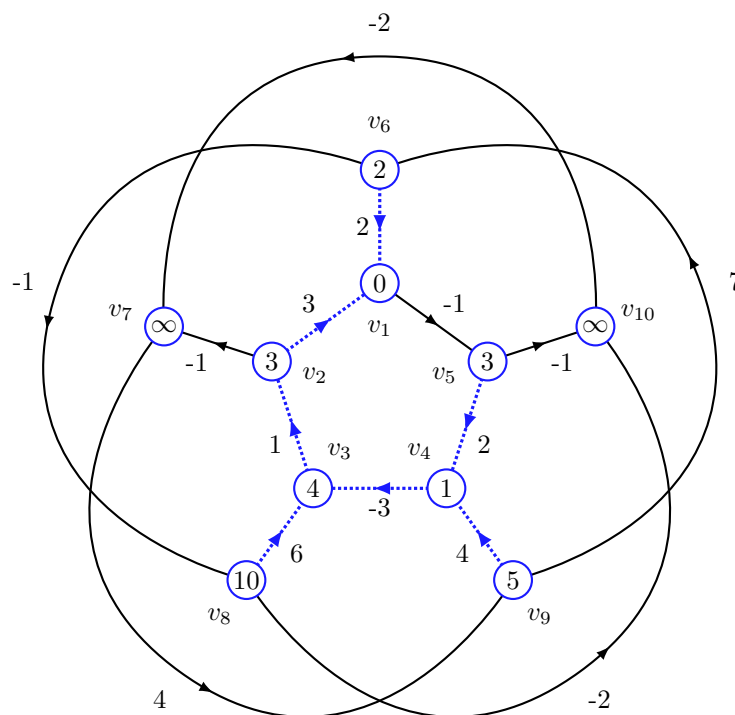


Let  $U$  be the 2-3-4 tree formed by first using the **Insert** operation to add 4 into  $T$ , then using the **Delete** operation to remove 19 from  $T$ .

- (a) What is the path traversed on applying the **Find** operation to search for 3 in  $U$ ? (For example, in  $T$ , this would be (9 13 17), (5 7), (1 2 3).) (2 marks)
- (b) What is the path traversed on applying the **Find** operation to search for 16 in  $U$ ? (3 marks)

**Question 12** (10 marks)

(Medium question.) Consider the edge-weighted directed graph below, pictured part of the way through executing the Bellman-Ford algorithm to find the distances  $d(v, v_1)$  for all vertices  $v$ . The current bounds on distance recorded by the algorithm are written inside each vertex. The edges currently selected by the algorithm are drawn thicker, dotted, and in blue.



Carry out **one** further iteration of the Bellman-Ford algorithm — that is, updating each vertex exactly once — processing the vertices in the order  $v_1, v_2, \dots, v_{10}$ . After carrying out this iteration:

- What is the weight of  $v_5$ ? (2 marks)
- What is the weight of  $v_7$ ? (2 marks)
- What is the weight of  $v_{10}$ ? (2 marks)
- What is the currently-stored path from  $v_7$  to  $v_1$ ? (2 marks)
- Is another iteration of the algorithm required to achieve accurate distances to  $v_1$ ? (2 marks)

**Question 13** (10 marks)

(Medium question.)

**Context:** You are a lowly apprentice to the master dwarven blacksmith Cheery Littlebottom, and you would like to use the facilities in your spare time to reforge the dread blade of A’Neem’Ay before the sacred festival of Jankon; this will allow you take over the world and avoid ever having to do Cheery’s laundry again. Unfortunately, while Cheery’s forge contains many anvils, most of them are in use most of the time, and as an apprentice you are unable to book any anvil time of your own — you must work in the gaps left by more important people. Worse, before you can make use of an anvil, you must spend a full day secretly dedicating it to A’Neem’Ay, and only one anvil can

(cont.)

be dedicated this way at once. The one piece of good news is A'Neem'Ay has given you the ability to see the future, and you know in advance how much time you'll be able to get at each anvil on each day.

**Problem:** You are given a set of  $k$  anvils  $A_1, \dots, A_k$ , and a time limit of  $D$  days. For each anvil  $A_i$ , you are also given a length- $D$  list  $h_i$  (indexed from 1) such that  $h_i[j] \in \{1, \dots, 16\}$ . For all  $j \in [D]$ , on day  $j$ , if you are currently at anvil  $i$ , then you can **either** spend  $h_i[j]$  hours working at anvil  $A_i$ , **or** spend the full day switching to a different anvil. Your goal is to maximise your total time spent working, across all anvils.

**Fill in the blanks** of the following dynamic programming algorithm, which solves the problem in  $O(kD)$  time. **Hint:** This algorithm makes use of a space-saving technique that was also used in the Bellman-Ford algorithm.

(The first four blanks are worth two marks each, the last two are worth one mark each.

**Don't copy the whole algorithm out**, just write what should go in each blank!)

---

**Algorithm:** WORLDCONQUEST

---

```
1 Let actions[i] ← "" for all  $i \in [k]$ .
2 Let hours[i], temp[i] ← 0 for all  $i \in [k]$ .
3 for  $j = 1$  to  $d$  do
4     for  $i = 1$  to  $k$  do
5         | temp[i] ← hours[i].
6     for  $i = 1$  to  $k$  do
7         | Let foo ← _____ + _____.
8         | Let bar ← _____{temp[x]:  $x \in [k]$ }.
9         | Let  $\ell$  be such that temp[ $\ell$ ] = _____.
10        | if foo > bar then
11            | | hours[i] ← foo.
12            | | Add "Work on the sword. " to the _____ of actions[i].
13        | else
14            | | hours[i] ← bar.
15            | | Add "Switch to anvil  $\ell$ . " to the _____ of actions[i].
16 for  $i = 1$  to  $k$  do
17     | Add "Start at anvil  $i$ . " to the start of actions[i].
18 Let  $\ell$  be such that hours[ $\ell$ ] = max{hours[i]:  $i \in [k]$ }.
19 Return actions[ $\ell$ ].
```

---

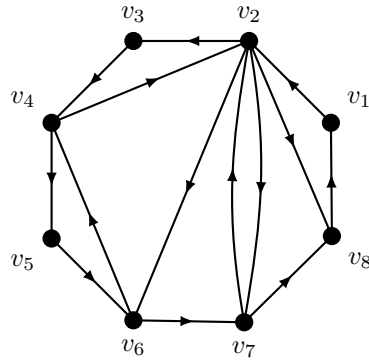


## Section 2 — Long-answer questions (75 marks)

In this section, you should give the reasoning behind your answers unless otherwise specified — you **will** receive credit for partial answers or for incorrect answers with sensible reasoning.

### Question 14 (10 marks)

(Short question.) Consider the directed graph  $G$  below.



- Express  $G$  in adjacency matrix form. (5 marks)
- Express  $G$  in adjacency list form. (5 marks)

### Question 15 (15 marks)

(Short question.)

**Context:** You are mayor and administrator of the thriving dwarven settlement of Bronzemurder. A series of trade delegations have arrived, and you have goods to sell. In fact, you have a *lot* of goods to sell — more than the traders can carry back with them — and you would like to extract as much immediate profit from the situation as possible. (The fortress has recently struck adamantine, and you have heard rumours that this often ends poorly in the long run.)

**Problem:** You are given the following information:

- A set  $\{T_1, \dots, T_k\}$  of traders;
- A set  $\{G_1, \dots, G_n\}$  of types of goods available;
- The total weight  $c_i$  of goods that each trader  $T_i$  is able to carry;
- The weight  $w_j$  of one unit of each goods type  $G_j$ ;
- The number  $n_j$  of units of each goods type  $G_j$  you have available to sell;
- The price  $p_{i,j}$  that trader  $T_i$  is willing to pay for one unit of goods of type  $G_j$ .

You wish to maximise your total profit by selling your goods to the appropriate traders. Express this as a linear programming problem, under the assumption that a single unit of goods can be split into fractional parts and sold to different traders. **Briefly** explain what your variables represent, and the purpose of each line of your answer.

**Question 16** (15 marks)

Consider the following problem, called FAULTYSAT. You are given a formula  $F$  in CNF form with  $3m$  clauses. You wish to know whether it is possible to satisfy not  $F$  itself, but at least  $2m$  of its clauses. For example, if  $F = (\neg x) \wedge (\neg y) \wedge (x \vee y)$ , then  $F$  itself is unsatisfiable, but still a Yes instance of FAULTYSAT (on setting  $x$  and  $y$  to False).

- (a) (Short question.) Prove that FAULTYSAT is in NP. (5 marks)
- (b) (Medium question.) Prove that FAULTYSAT is NP-complete. (10 marks)

**Question 17** (10 marks)

(Medium question.) John is trying to come up with an algorithm to test whether two **connected** graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are isomorphic in polynomial time, and comes up with the following.

---

**Algorithm:** BADISOTEST

---

```

1  for  $d = 0$  to  $n$  do
2    if  $G_1$  doesn't contain the same number of degree- $d$  vertices as  $G_2$  then
3      Return No.
4  Let  $\Delta$  be the maximum degree of  $G_1$ .
5  Let  $v$  be an arbitrarily-chosen vertex of degree  $\Delta$  in  $G_1$ .
6  Using BFS, let  $L_i(v) = \{x \in V_1 : d(v, x) = i \text{ in } G_1\}$  for all  $i \in [n]$ .
7  for  $w \in V_2$  of degree  $\Delta$  in  $G_2$  do
8    Using BFS, let  $L_i(w) = \{x \in V_2 : d(w, x) = i \text{ in } G_2\}$  for all  $i \in [n]$ .
9    if for all  $i, d \in [n]$ ,  $L_i(v)$  contains the same number of degree- $d$  vertices in  $G_1$  as  $L_i(w)$  contains in  $G_2$  then
10     Return Yes.
11 Return No.
```

---

Give two connected graphs  $G_1$  and  $G_2$  which are not isomorphic, but which have the property that  $\text{BADISOTEST}(G_1, G_2)$  is guaranteed to incorrectly return Yes, no matter how  $v$  is chosen. **Briefly** explain why  $\text{BADISOTEST}(G_1, G_2)$  always returns Yes. (You do not have to explain why  $G_1$  and  $G_2$  are not isomorphic.)

**Question 18** (5 marks)

(Medium question.) Let  $G$  be a graph, let  $M$  be a matching in  $G$  of size 50, and let  $M^*$  be a **maximum** matching in  $G$ . Suppose that the symmetric difference of  $M$  and  $M^*$  contains exactly 6 components. What are the possible sizes of  $M^*$ ? **Briefly** explain your answer. (**Hint:** You may find it helpful to use ideas from the proof of Berge's lemma.)

**Question 19** (5 marks)

(Long question.)

**Context:** The war is over, the forces of good have fallen, and Dread Emperor Triumphant stands, well, triumphant. The opposition was led by the Rogue Sorcerer, and the Dread Emperor has decreed that there will be a grand victory parade from the site of her final defeat to the newly-rechristened Imperial Palace. As the Dread Emperor's most senior viceroy, you have been put in charge of planning the route.

Unbeknownst to Triumphant, you are a double agent for the forces of good plotting his assassination. Rebels have set up traps in several isolated streets in the city, and

you would like to lead the parade through as many of them as possible. To minimise suspicion, however, you plan to ensure that every street the parade passes through will take it closer to the Palace rather than further away (or the same distance).

**Problem:** You are given a set of points  $P$ , and a set of streets  $S$  joining pairs of points. You are given two points  $D, IP \in P$ , and a set of streets  $T \subseteq S$ . For each point  $p \in P$ , you are given the distance  $d_p$  of  $p$  from  $IP$  (in a straight line). You wish to find a path  $x_0 \dots x_t$  along the streets in  $S$  such that:

- $x_0 = D$ ;
- $x_t = IP$ ;
- for all  $i \in \{1, \dots, t\}$  we have  $d_{x_i} < d_{x_{i-1}}$ ;
- subject to the above,  $x_0 \dots x_t$  passes through as many streets in  $T$  as possible.

Give an algorithm to solve this problem in polynomial time, and **briefly** explain why it works. (**Hint:** Try reducing this problem to one you already have an algorithm for.)

**Question 20** (5 marks)

(Long question.) We say a *covering set* for a graph  $G = (V, E)$  is a set  $X \subseteq V$  such that every vertex in  $G$  is either in  $X$  or adjacent to some vertex in  $X$ , i.e. such that  $V = X \cup N(X)$ . (Recall that  $N(X) = \bigcup_{v \in X} N(v)$ .) You are interested in finding as **small** a covering set as possible in an  $n$ -vertex graph  $G$ , and come up with the following greedy algorithm: Initially, take  $X = \emptyset$ . Then, repeatedly add vertices of maximum degree in  $G[V \setminus (X \cup N(X))]$  to  $X$  until  $X \cup N(X) = V$ , i.e. until  $X$  is covering. Give an example to show that the covering set this algorithm finds might be  $\Omega(n)$  vertices larger than a minimum covering set, and **briefly** explain why your example works.

(You may choose how the algorithm breaks ties — that is, whenever there are multiple vertices of maximum degree in  $G[V \setminus (X \cup N(X))]$ , you may choose which of them the algorithm adds to  $X$ .)

**Question 21** (5 marks)

(Long question.) Let  $G_n$  be the graph whose vertex set is the set of all trees with vertex set  $[n]$ . (So the vertices of  $G_n$  are trees.) Two  $n$ -vertex trees  $T_1$  and  $T_2$  are joined by an edge in  $G_n$  if  $T_2$  can be formed from  $T_1$  by removing one edge, then adding one edge. Prove that  $G_n$  is connected.

**Question 22** (5 marks)

(Long question.) You are running a polyamorous dating site, and are finding that the problem of finding compatible groups seems to be harder to solve than finding large matchings in a bipartite graph. Consider the following highly simplified version of the problem, in which each person is only interested in a closed relationship with exactly two others. You are given a set  $S$  of people, and a set  $T$  of compatible triples of people. We say  $D \subseteq T$  is a *dating arrangement* if  $t_1 \cap t_2 = \emptyset$  for all distinct  $t_1, t_2 \in D$ ; you seek a dating arrangement which is as large as possible. Prove that even this simplified version

(cont.)

of the problem is still NP-hard to solve exactly. You don't need to spell out every detail of the argument, but it should be clear how and why your reduction works.

**Hint:** Try reducing from 3-SAT. You will find gadgets similar to the ones shown below very useful, where the dots represent people and the triangles represent triples in  $T$ .

