UNIVERSITY OF BRISTOL

Not a Real Examination Period

Department of Computer Science

2nd Year Practice Paper 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

This paper contains eighteen questions.
All questions will be marked except where otherwise indicated.
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 bring up to four A4 sheets of pre-prepared notes with you into the exam, but no other written materials.**
2. **You may use a calculator with the Faculty seal of approval if you wish.**

TURN OVER ONLY WHEN TOLD TO START WRITING

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

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

(a) $n \in \Omega(\sqrt{n})$. (1 mark)

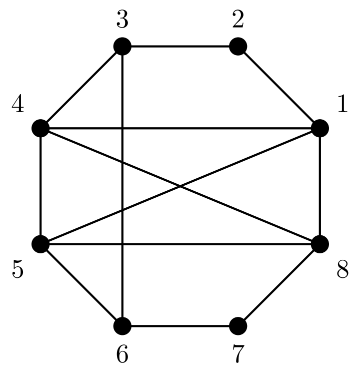
(b) $n \in o(100n)$. (1 mark)

(c) $n \in O(100n)$. (1 mark)

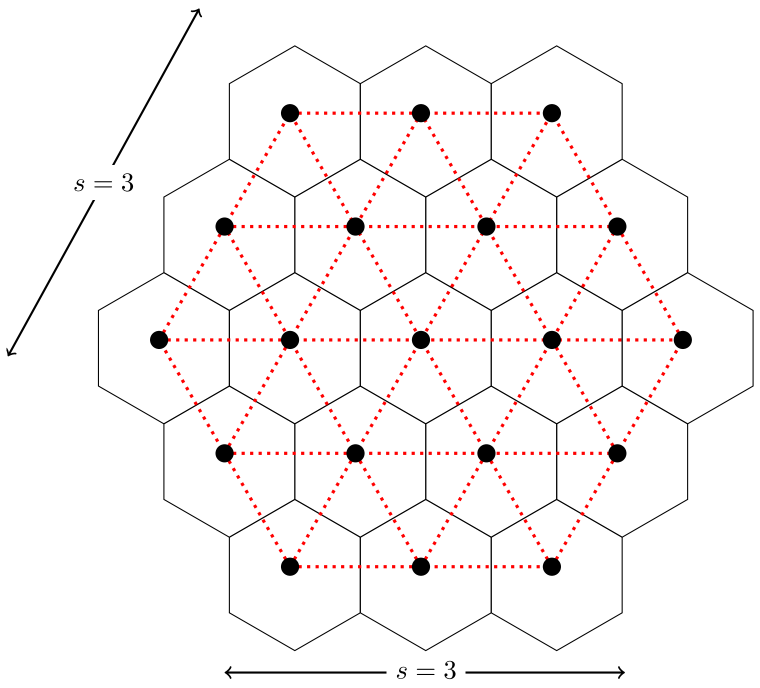
(d) $\log n \in O(n^{1/100})$. (1 mark)

(e) $(\log n)^{\log n} \in O(n)$. (1 mark)

Question 2 (5 marks)
Write down an Euler walk for the following graph.

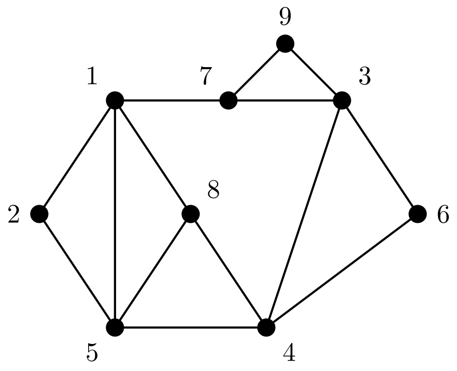


Question 3 (5 marks)
Let $s \geq 2$ be an integer. Consider a regular hexagonal arrangement of regular hexagonal cells, with each side consisting of s cells. Consider the graph G_s formed by taking each cell to be a vertex, and joining two cells by an edge if they share a side. An example is shown below for $s = 3$, where the black lines show the arrangement of cells and the red dotted lines show the edges of G_s.



Give a formula for the number of edges in G_s in terms of s. You may use the fact that G_s has $3s^2 - 3s + 1$ vertices. (You do not need to show your working — only your final answer will be marked. You may wish to check that your answer is correct when $s = 2$.)

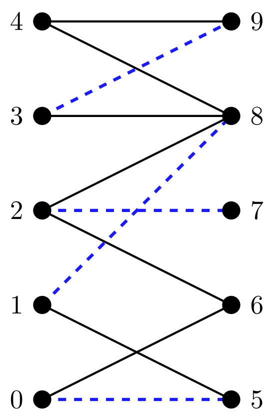
Question 4 (5 marks)
Consider a depth-first search in the following graph starting from vertex 1.



In the implementation of depth-first search given in lectures, we say vertex i is *explored* when `explored`$[i]$ is set to 1. (For example, the start vertex is explored first.) Which vertex will be explored sixth if the start vertex is 4? 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)

Consider the graph G and the matching $M \subseteq E(G)$ shown below, where M is drawn with blue dashed lines.



Write down an augmenting path for M in G.

Question 6 (5 marks)

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`([8]);
`Union`(1, 2);
`Union`(1, 3);
`Union`(3, 4);
`Union`(5, 6);
`Union`(5, 1);
`Union`(7, 8);
`Union`(6, 7).

(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 7 (5 marks)

Let $\mathcal{G} = (G, c_E, c_V, s, t)$ be a vertex flow network with $|V(G)| = 100$. In lectures, we covered a way of finding a maximum flow in $\mathcal{G}$ by applying the Ford-Fulkerson algorithm to a new flow network (H, c, s', t'). How many vertices will H have, in this case? Choose **one** of the following options.

A. 98.

B. 100.

C. 102.

D. 198.

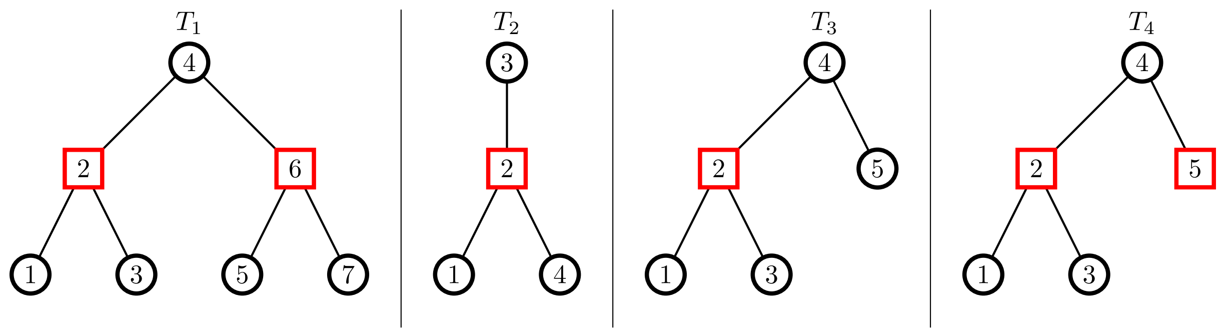
E. 200.

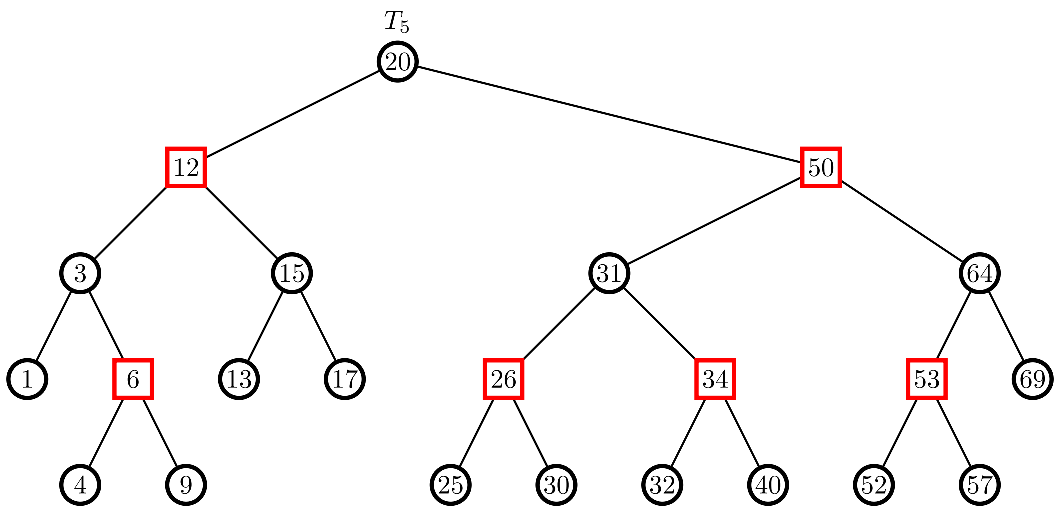
F. 202.

G. None of the above, or it's impossible to tell.

Question 8 (5 marks)

Which of the following trees $T_1, \ldots, T_5$ are valid red-black trees? (In case you are unable to distinguish the colours, the red nodes are drawn as squares and the black nodes are drawn as circles.)





Question 9 (5 marks)

Consider an instance of interval scheduling with interval set

$$\mathcal{R} = \{(1,3), (2,6), (4,9), (5,6), (6,11), (7,8), (9,11), (10,12), (11,12), (11,13)\}.$$

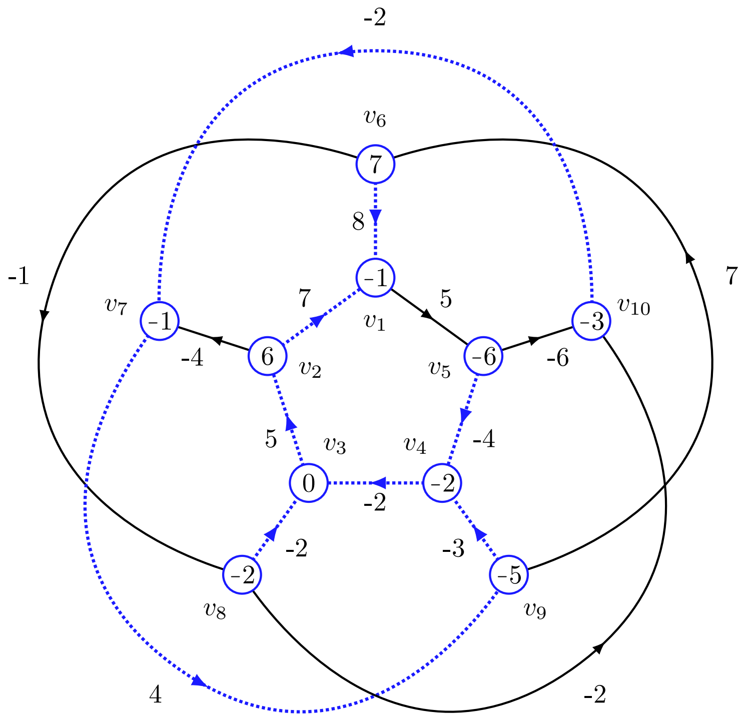
(a) When the greedy interval scheduling algorithm discussed in lectures is run on this input, which interval will it choose third? (3 marks)

(cont.)

(b) What is the size of a maximum compatible set? (2 marks)

Question 10 (10 marks)

Consider the edge-weighted directed graph below, pictured part of the way through executing the Bellman-Ford algorithm to find the distances $d(v, v_3)$ for all vertices v, i.e. the single-**sink** version of the algorithm with sink v_3. 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 $v_1, v_2, \ldots, v_{10}$. After carrying out this iteration:

(a) What is the weight of v_2? (2 marks)

(b) What is the weight of v_6? (2 marks)

(c) What is the weight of v_{10}? (2 marks)

(d) What is the currently-stored path from v_5 to v_3 (again, **after** carrying out the iteration)? (2 marks)

(e) Is another iteration of the algorithm required to achieve accurate distances to v_1? (2 marks)

Question 11 (10 marks)

You are trying to get a high score in the popular video game Tambourine Hero. In this game, you score points by shaking your tambourine in time to a song playing in the background. After playing certain beats of the song, you are awarded "star power"; your stored star power is an integer between 0 and 100. At any time when you have 50 or more star power, you can "activate star power". You will then lose star power at a rate of one point per beat until you run out, at which point it is deactivated; while star power is activated, you will earn double points. Any extra star power you are awarded over 100 points is wasted.

Formally, a *song* is a series of *beats* $b_1, \dots, b_t$. Each beat b_i is a pair (p_i, s_i) of p_i *points* and s_i *star power*, where p_i and s_i are non-negative integers. Let P_i be your total points after beat i, and let S_i be your total star power after beat i. Initially, $P_0 = S_0 = 0$. Subsequently, after beat $i \geq 1$,

$$P_i = \begin{cases} P_{i-1} + 2p_i & \text{if star power active,} \\ P_{i-1} + p_i & \text{otherwise.} \end{cases}$$

$$S_i = \begin{cases} \min(100, S_{i-1} + s_i) - 1 & \text{if star power active,} \\ \min(100, S_{i-1} + s_i) & \text{otherwise.} \end{cases}$$

After each beat, if $S_i \geq 50$ and star power is not active, you may choose to activate star power before the next beat. It then stays active until the end of the next beat i with $S_i = 0$. **Fill in the blanks** in the following dynamic programming algorithm, which outputs a list of the beats on which to activate star power in order to achieve the maximum possible score.

(**Don't copy the whole algorithm out**, just write what should go in each blank! Each blank should contain a single expression, e.g. $\max(b, c)$ or `next`$[b][s][a]$. Two marks will be awarded per blank correctly filled in.)

```
Algorithm: BeATambourineHero
1  Let next[b][s][a], score[b][s][a] ← 0 for all $0 \leq b \leq t$, all $0 \leq s \leq 100$, and all $a \in \{0,1\}$.
2  for $b = t - 1$ to 0 do
3      for $s = 1$ to 100 do
4          Let new_s_inactive ← ______.
5          Let inactive_score ← $p_b$ + score[$b+1$][new_s_inactive][0].
6          Let activating_score ← $p_b$ + score[$b+1$][new_s_inactive][1] if $s \geq 50$, and activating_score ← $-1$
             otherwise.
7          Let score[$b$][$s$][0] ← max(inactive_score, activating_score).
8          Let score[$b$][$s$][1] ← $2p_b$ + score[$b+1$][new_s_inactive − 1][______] if new_s_inactive $\geq 2$, and
             score[$b$][$s$][1] ← $2p_b$ + score[$b+1$][0][0] otherwise.
9          If activating_score > inactive_score, let next[$b$][$s$][0] = 1.
10         If new_s_inactive $\geq 2$, let next[$b$][$s$][1] = 1.
11 Let active ← 0 and $s$ ← 0.
12 for $b = 0$ to $t - 1$ do
13     If ______ = 0 and ______ = 1, print "Activate star power after beat $b$".
14     Let $s \leftarrow \min(100, s + s_{b+1}) - 1$ if active = 1 and $s \leftarrow \min(100, s + s_{b+1})$ otherwise.
15     Let active ← next[$b+1$][$s$][______].
```

Question 12 (5 marks)

A *Hamilton path* in a graph G is a path containing every vertex, i.e. a Hamilton cycle minus an edge. If G is a graph and $x, y \in V(G)$, we define HP(G, x, y) to be the problem of deciding whether or not G contains a Hamilton path from x to y, so (G, x, y) is a `Yes` instance if such a path exists and a `No` instance otherwise. Likewise, we define HC(G) to be the problem of deciding whether or not G contains a Hamilton cycle.

(a) Is it true that both HP and HC are decision problems? (1 mark)

(b) Is it true that both HP and HC are in NP? (2 marks)

(c) Is it true that if we require every vertex in the input graph G to have degree at least $|V(G)|/2$, then HC is in P? (You may assume P $\neq$ NP.) (2 marks)

Question 13 (5 marks)
Consider the following reduction between the problems HC and HP introduced in Question 12. Let G be an instance of HC. For every edge $\{x, y\} \in E(G)$, run an algorithm (or oracle) for HP with inputs $G - \{x, y\}$, x, and y, where $G - \{x, y\}$ is the graph formed from G by removing $\{x, y\}$ from $E(G)$ and leaving $V(G)$ unchanged. If any of those algorithms output `Yes`, then return `Yes`; otherwise, return `No`.

(a) Is this a reduction from HC to HP, or a reduction from HP to HC? (3 marks)

(b) Is this a Karp reduction, or a Cook reduction? (2 marks)

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

Question 14 (5 marks)
Give an example of a graph which contains a length-5 cycle as a subgraph, but not as an induced subgraph.

Question 15 (15 marks)
You are running a long-haul trucking company. You have a fixed number of vehicles which transport material between a set of cities $C_1, \dots, C_k$. For all $i \in [k]$, you currently have $t_i \geq 0$ trucks in city C_i. Over the course of the day, each truck in city C_i can haul a load to any other city C_j with total profit $p_{i,j}$ (after accounting for fuel costs and so on). Each city contains a SimplifyTheProblem Inc. depot at which your trucks all stop. These depots handle loading, unloading, and various administrative tasks, but the depot in city C_i is only under contract to receive and unload $T_i \geq 1$ trucks per day; thus you must avoid sending more than T_i trucks to city C_i. Assume that between travel, loading, and unloading, each trip between any pair of cities takes the full day.

Your goal is to choose destinations for your trucks to maximise your total profit for today without any regard for the future. (Coincidentally, you also invest heavily in fossil fuels and cryptocurrencies.) Formulate this as a linear programming problem and give a **brief** explanation of what your variables represent and why your constraints and objective function are appropriate.

Question 16 (10 marks)
Explain briefly why the reduction between HC and HP of Question 13 is valid. (A good answer here will likely be no longer than one paragraph, and certainly no longer than two paragraphs.)

Question 17 (30 marks)
Solve two from the following four questions (15 marks each). **If you attempt more than two, you will not gain any extra credit, and only the first two questions attempted will be marked. If you do not wish one of your attempts to be marked, cross it out clearly along with all working.**

(a) 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 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.)

(b) Let $G = (V, E)$ be a connected graph with edge weights given by $w \colon E \to \mathbb{R}$. You may assume that every edge gets a different weight. Let C be a cycle in G, and let e be the highest-weight edge in C. It can be shown that no minimum spanning tree of G contains e.

Using this fact or otherwise, give an algorithm which, given an n-vertex connected graph $G = (V, E)$ in adjacency list format with $|E| = n + 50$, outputs a minimum

(cont.)

spanning tree in $\boldsymbol{O(n)}$ time. Briefly explain why your algorithm works and why it runs in $O(n)$ time.

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

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

(d) In this question, we work with a variant of SAT in which variables cannot be negated. Given literals a, b and c, which need not be distinct, an *even clause* EVEN(x, y, z) evaluates to `True` if and only if either zero or two of x, y and z evaluate to `True`. A *width-3 positive OR clause* is an OR clause of three variables (i.e. **un-negated** literals). A *positive even formula* is a conjunction of even clauses and width-3 positive OR clauses. For example,

$$\text{EVEN}(a, \neg b, c) \wedge \text{EVEN}(a, a, d) \wedge (a \vee b \vee e) \wedge \text{EVEN}(c, d, e).$$

is a positive even formula, but $(\neg a \vee b)$ is not due to both the negated variable and the fact that the clause only contains two variables. The decision problem POS-EVEN-SAT asks whether a positive even formula (given as the input) is satisfiable, in which case the desired output is `Yes`.

i. Give a Karp reduction from POS-EVEN-SAT to 3-SAT and briefly explain why it works.

ii. Give a Karp reduction from 3-SAT to POS-EVEN-SAT and briefly explain why it works.

Question 18 (15 marks)

Choose <u>one</u> of the three following problems to solve. **If you attempt more than one, you will not gain any extra credit, and only the first question attempted will be marked. If you do not wish one of your attempts to be marked, cross it out clearly along with all working.**

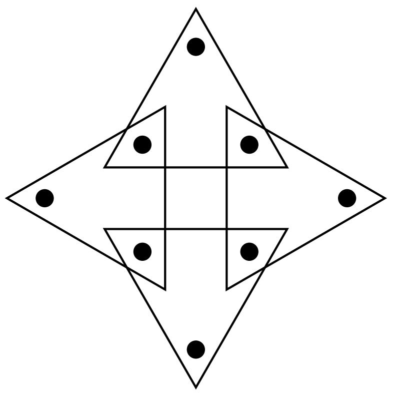
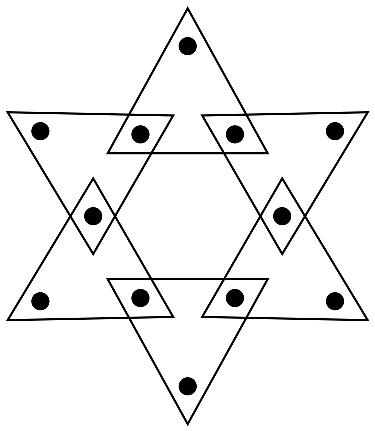
(a) For any connected graph G, we say that a vertex $v \in V(G)$ is *outside the core* if $G - v$ is connected. Prove that any connected graph G with at least two vertices contains at least one vertex outside the core. (One possible approach uses induction.)

(b) Consider a barter economy with goods $G_1, \dots, G_t$. For all i and j, you can directly trade x_i units of G_i for y_j units of G_j; this is expressed as the ratio $r_{i,j} = x_i / y_j$. Notice that you can trade one unit of G_i for $r_{ij} r_{jk}$ units of G_k, by first trading G_i for G_j and then trading G_j for G_k; the same holds for longer chains of trades. You are interested in becoming obscenely wealthy, and so you are looking for a *trading cycle* $C = x_{i_1} \dots x_{i_t} x_{i_1}$ such that the product $r_{i_1,i_2} \dots r_{i_{t-1},i_t} r_{i_t,i_1}$ of all the ratios is strictly greater than 1, leaving you with more goods G_1 than you started with. We call such a cycle an *arbitrage cycle.* Give a polynomial-time algorithm to decide whether an arbitrage cycle exists, and explain why it works.

(**Hint:** Recall that distances are not well-defined in a directed graph with cycles of negative total weight. In fact, you can check whether such cycles exist in polynomial time by running the Bellman-Ford algorithm for one extra iteration and seeing whether the weights change — if they do, then the graph contains a negative-weight cycle.)

(c) 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 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.

END OF PAPER