UNIVERSITY OF BRISTOL

August 2022 Reassessment 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.

You are strongly advised to attempt questions tagged as "short" before questions tagged as "medium", as they typically require less time for more marks. Be aware that Section 2 contains "short" questions as well.

Question 1 (5 marks)

(Short question.) 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)

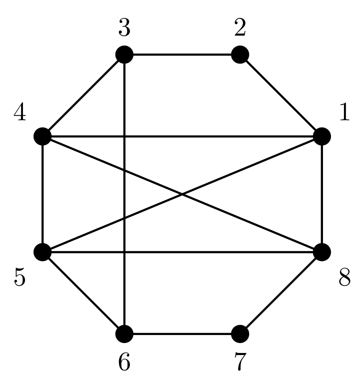
(c) $n \in O(100n)$. (1 mark)

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

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

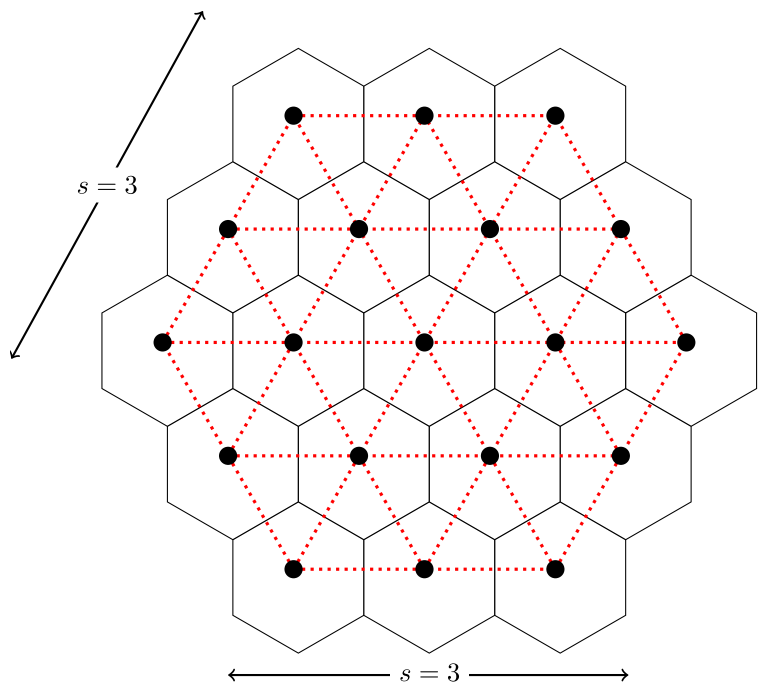
Question 2 (5 marks)

(Short question.) Write down an Euler walk for the following graph.



Question 3 (5 marks)

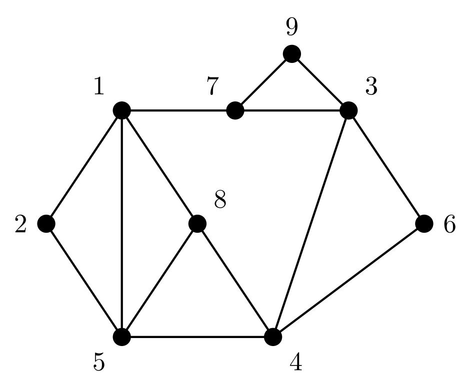
(Medium question.) Let $s \geq 2$ be an integer. Consider a regular hexagonal arrangement of regular hexagonal cells, with each side consisting of s cells. Consider the graph G_s formed by taking each cell to be a vertex, and joining two cells by an edge if they share a side. An example is shown below 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)

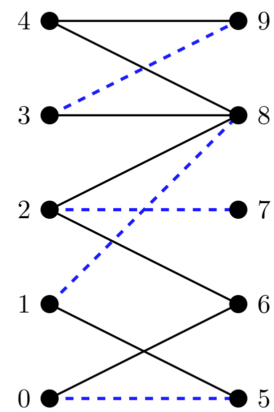
(Short question.) Consider a depth-first search in the following graph starting from vertex 1.



In the implementation of depth-first search given in lectures, we say vertex i is *explored* when `explored`$[i]$ is set to 1. (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)

(Short question.) 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)

(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`([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)

(Short question.) 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.

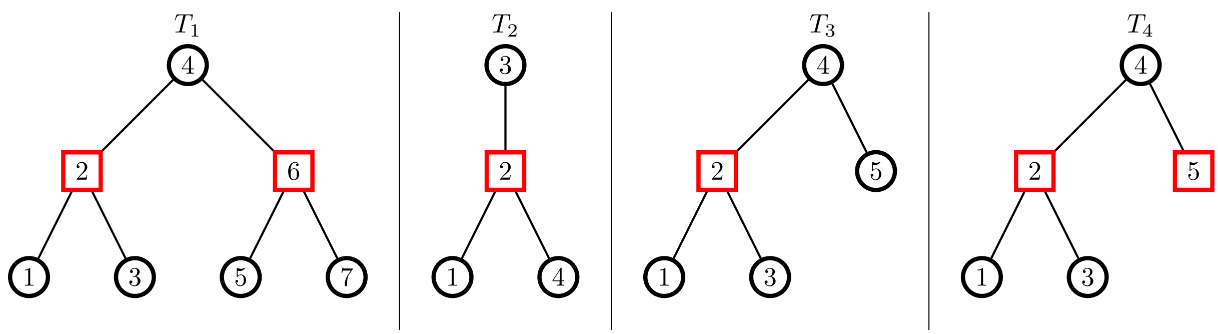
E. 200.

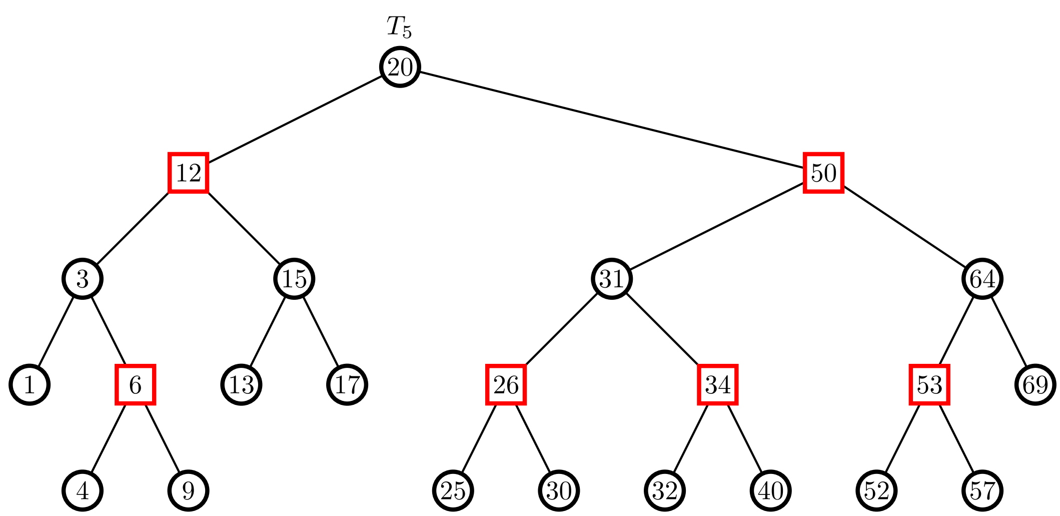
F. 202.

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

Question 8 (5 marks)

(Short question.) 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)

(Short question.) Consider an instance of interval scheduling with interval set

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

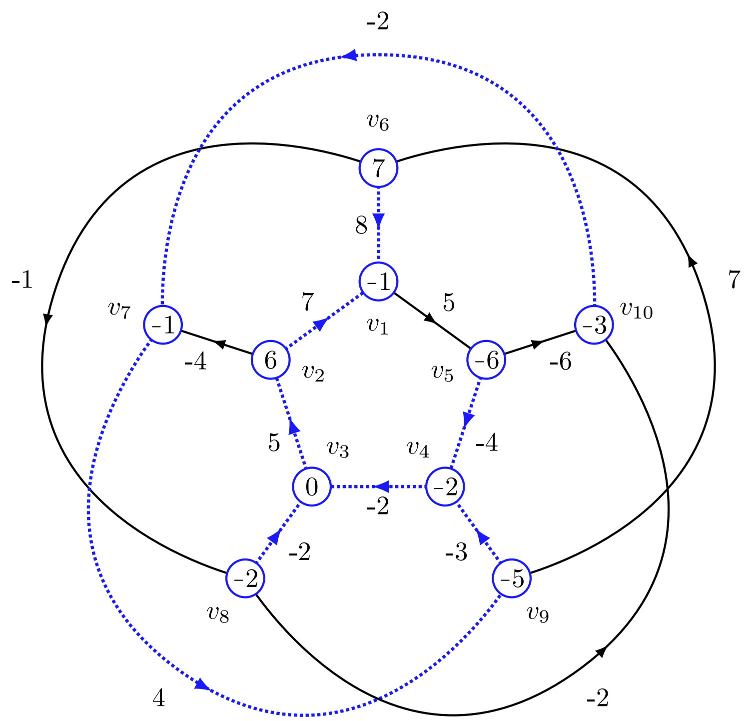
(a) When the greedy interval scheduling algorithm discussed in lectures is run on this input, which interval will it choose third? (3 marks)

(cont.)

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

Question 10 (10 marks)

(Medium question.) Consider the edge-weighted directed graph below, pictured part of the way through executing the Bellman-Ford algorithm to find the distances $d(v, v_3)$ 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)

(Medium question.) 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

```
Let next[b][s][a], score[b][s][a] ← 0 for all 0 ≤ b ≤ t, all 0 ≤ s ≤ 100, and all a ∈ {0,1}.
for b = t − 1 to 0 do
    for s = 1 to 100 do
        Let new_s_inactive ← ______.
        Let inactive_score ← p_i + score[b + 1][new_s_inactive][0].
        Let activating_score ← p_i + score[b + 1][new_s_inactive][1] if s ≥ 50, and activating_score ← −1
          otherwise.
        Let score[b][s][0] ← max(inactive_score, activating_score).
        Let score[b][s][1] ← 2p_i + score[b + 1][new_s_inactive − 1][______] if new_s_inactive ≥ 2, and
          score[b][s][1] ← 2p_i + score[b + 1][0][0] otherwise.
        If activating_score > inactive_score, let next[b][s][0] = 1.
        If new_s_inactive ≥ 2, let next[b][s][1] = 1.

Let active ← 0.
for b = 0 to t − 1 do
    If ______ = 0 and ______ = 1, print "Activate star power after beat b".
    Let active ← next[b + 1][s][______].
```

Question 12 (5 marks)

(Short question.) A *Hamilton path* in a graph G is a path containing every vertex, i.e. a

(cont.)

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)

(Short question.) 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)

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.

You are strongly advised to attempt questions tagged as "short" before questions tagged as "medium", and questions tagged as "medium" before questions tagged as "long", as they typically require less time for more marks.

Question 14 (5 marks)

(Medium question.) 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 15 (15 marks)

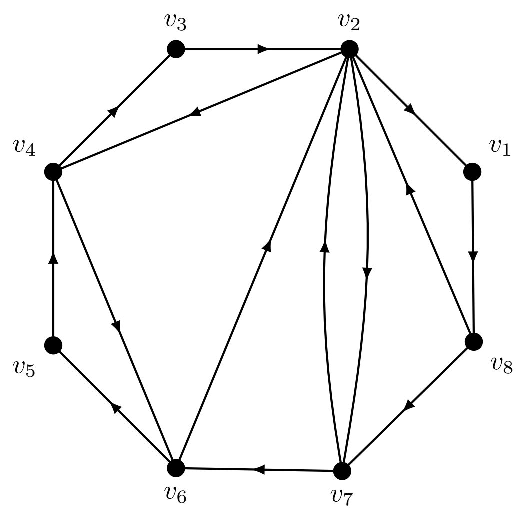
(Short question.) You are running a long-haul trucking company. You have a fixed number of vehicles which transport material between a set of cities $C_1, \dots, C_k$. For all $i \in [k]$, you currently have $t_i \geq 0$ trucks in city C_i. Over the course of the day, each truck in city C_i can haul 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)

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

(cont.)



(a) Express G in adjacency matrix form. (5 marks)

(b) Express G in adjacency list form. (5 marks)

Question 17 (5 marks)

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

Question 18 (15 marks)

(Medium question.) 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* $\text{EVEN}(x, y, z)$ 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`.

(a) Give a Karp reduction from POS-EVEN-SAT to 3-SAT and briefly explain why it works. (5 marks)

(b) Give a Karp reduction from 3-SAT to POS-EVEN-SAT and briefly explain why it works. (10 marks)

Question 19 (10 marks)

It is five minutes to midnight, and you are a secret agent trying to prevent a nuclear dead man's switch from destroying the world. In order to stop the bombs from going off, you have passcode audio files $C_1, \ldots, C_n$ which must be sent to a remote system exactly and without interruptions starting from midnight. For perfect fidelity, audio clip C_i must be sent at a rate of m_i megabits per second for t_i seconds. Unfortunately, you are working from the comic relief character's house, and their ISP implements throttling in an unpleasant way: if you send more than t megabits in the first t seconds of the day, your connection will go down for several minutes (thereby destroying the world). Fortunately, you can send the clips in any order, and not all of them have $m_i > 1$. Your goal is to save the world by sending all n audio files to the system at the specified bitrates, without gaps or interruptions from the ISP. You may assume this is always possible.

(a) (Medium question.) Prove that the greedy algorithm which sends a longest audio file first (i.e. a file with t_i maximum) **fails**, by giving a counter-example with a brief explanation. (5 marks)

(b) (Long question.) State a greedy algorithm which works, and prove that it works using either an exchange argument or a "greedy stays ahead" argument. (5 marks)

Question 20 (5 marks)

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

Question 21 (5 marks)

(Long question.) 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.)

Question 22 (5 marks)

(Long question.) In the problem ZERO-SUM-SET, you are given a positive integer k and a set X of integers and asked whether there exists a size-k subset of X which sums to zero. For example, $k = 3$ and $X = \{2, -3, 1, 4\}$ is a `Yes` instance of ZERO-SUM-SET since $2 - 3 + 1 = 0$, but it would be a `No` instance for $k = 2$.

The problem ZERO-SUM-LISTS is similar, but the input is k lists $X_1, \dots, X_k$ of integers rather than a single set, and you are asked whether you can choose one integer from each list such that their sum is zero — that is, choose $x_1, \dots, x_k$ with $x_i \in X_i$ for all i and

(cont.)

$\sum_i x_i = 0$. For example, $X_1 = \{2, 4, 5\}$, $X_2 = \{-4, -1, 5\}$, $X_3 = \{1, 2\}$ is a `Yes` instance of ZERO-SUM-LISTS because $2 - 4 + 2 = 0$, $2 \in X_1$, $-4 \in X_2$ and $2 \in X_3$.

Give a Karp reduction from ZERO-SUM-LISTS to ZERO-SUM-SET. (You may assume all arithmetic operations take constant time.)

END OF PAPER