CS 240 Final exam

□□□

TOTAL POINTS

50 / 60

QUESTION 1

Question 1 10 pts

1.1 (1) **2 / 2**

- ✓ **- 0 pts Correct**
- **- 2 pts** ABD
- **- 1 pts** ABD

1.2 (2) **2 / 2**

- ✓ **- 0 pts Correct**
- **- 1 pts** BCD
- **- 2 pts** BCD

1.3 (3) **2 / 2**

- ✓ **- 0 pts AB**
- **- 1 pts** AB
- **- 2 pts** AB

1.4 (4) **2 / 2**

- ✓ **- 0 pts ABCD**
- **- 1 pts** ABCD
- **- 2 pts** ABCD

1.5 (5) **1 / 2**

- **- 0 pts** BCD
- ✓ **- 1 pts BCD**
- **- 2 pts** BCD

QUESTION 2

2 Question2 10 / 10

- ✓ **- 0 pts Correct**
- **- 2 pts** Wrong Complexity
- **- 2 pts** Wrong Computation Analysis
- **- 6 pts** Wrong Division
- **- 9 pts** Wrong Understanding

QUESTION 3

3 Question3 6 / 10

- **- 0 pts** Correct
- **- 10 pts** □□□□□□□□□
- **- 2 pts** □□□□□□□□□□□□□
- **- 4 pts** □□□students□□□□□□□□□□□□□□□□□□□□□□□□□1□□□□□□□□□infinity
- **- 9 pts** □□□□□□□□□□□□□no two students from the same university are in the same room
- ✓ **- 4 pts □□□students□□□□□□□□□□□□□□no two students from the same university are in the same room**
- **- 0.5 pts** source□S□□S□U□capacity□□□□1□□□□□□□□S□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□student□□□□□□□□□
- **- 4 pts** □□university□ U-R□capacity□□1
- **- 4 pts** university-room□capacity□□□□□□□□□□□□no two students from the same university are in the same room□□□
- **- 4 pts** □□□□□□□□return error message□□□□□□□□
- **- 4 pts** □□□□□□□□□□□i==0□□□□□□□□□□□□□□□□□□□□□□□
- **- 6 pts** □□□□□□□capacity□node□
- **- 10 pts** q<p□□□□□□□□□□□□□□□□□□□□□□□□□q□□□□□□p□□□□□□
- **- 2 pts** revised
- **- 4 pts** □□□□□□□□□□

QUESTION 4

4 Question4 6 / 10

- **- 0 pts** Correct
- **- 6 pts** The dynamic equation is not correct
- ✓ **- 2 pts Boundary conditions is not correct**
- ✓ **- 2 pts The time complexity is not correct**
- **- 10 pts** wrong or empty
- **- 6 pts** The algorithm is not optimal.

QUESTION 5

5 Question5 **10 / 10**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect reduction direction

- 2 pts Incorrect Construction

- 1 pts Partial Incorrect Construction

- 3 pts Incorrect get:

If exists a Hamilton Cycle, then we can get a tsp.

- 1.5 pts Partial Incorrect get:

If exists a Hamilton Cycle, then we can get a tsp.

- 3 pts Incorrect get:

If there exists a polynomial time k approximation algorithm A for TSP, then Hamilton Cycle Problem is P.

- 1.5 pts Partial Incorrect get:

If there exists a polynomial time k approximation algorithm A for TSP, then Hamilton Cycle Problem is P.

- 10 pts Incorrect

- 1 pts Show General TSP instead of K-approximate

QUESTION 6

6 Question6 **9 / 10**

+ 10 pts Correct

Algorithm

✓ **+ 2 pts** **Algorithm is based on multiple AB and C by vector**

✓ **+ 1 pts** **Algorithm is a Monte Carlo randomized algorithm**

+ 1 pts Specify the distribution of variable

+ 4 pts All correct

✓ **+ 1 pts** **O(n2) complexity**

✓ **+ 5 pts** **Omega(1) probability with valid proof**

+ 0 pts Wrong

CS 240 Algorithm Design and Analysis (Spring 2019)
Final Exam

Name (in Chinese): 吴天元

ID#: 63305667

Instructions

- Time: 1:00-2:40pm (100 minutes)
- This exam is closed-book, but you may bring an A4-size cheat sheet. Put all the study materials and electronic devices into your bag and put your bag in the front, back, or sides of the classroom.
- You can write your answers in either English or Chinese. You can use both sides of the paper.
- Two blank pieces of paper are attached on the back, which you can use as scratch paper. Raise your hand if you need more paper.

1 (10 pt)

Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 1 point if you select a non-empty proper subset of the correct answers.

1	2	3	4	5
ABD.	BCD	AB	A~~BC~~D	D.

ABD 1. Which of the following is correct?

A. $n^{\sqrt{n}}$ is $O(2^n)$ $n \log 2.$ $\sqrt{n} \log n.$ $\frac{n \log 2}{\sqrt{n} \log n} =$

B. 100^{100} is $O(\log n)$

C. $(\log n)^n$ is $O(n^{\log n})$ $n \log\log n \to \log n \log n.$

D. n^2 is $O((\log n)^{\log n})$ $2 \log n.$ $\log n \cdot \log\log n.$

E. None of above

BCD 2. Which of the following is known to be correct?

A. 3-SAT $\in$ P

B. 3-SAT $\in$ NP

C. 3-SAT $\in$ NP-complete

D. 3-SAT $\in$ NP-hard

E. None of the above.

AB 3. Which of the following is known to be correct?

A. $X \in \mathrm{P} \Rightarrow X$ has a poly-time certifier ✓

B. $X \in \mathrm{NP} \Rightarrow X$ has a poly-space certifier ✓

C. $X \in \mathrm{PSPACE} \Rightarrow X$ does not have a poly-time certifier ✗

D. $X \in$ PSPACE-complete $\Rightarrow X$ does not have a poly-time certifier ✗

E. None of above ✗

CABD 4. Which of the following is known to be correct?

A. 2-COLOR $\leq_p$ 3-COLOR

B. 2D-Matching $\leq_p$ 3D-Matching

C. 2D-Matching $\leq_p$ 2-COLOR

D. 3D-Matching $\leq_p$ 3-COLOR

E. None of the above.

AD 5. Which of the following is known to be correct?

A. If $X \in \mathrm{NP} \cap$ co-NP, then $X \in \mathrm{P}$

B. If $X \in \mathrm{NP}$, then its complement $\bar{X} \in$ co-NP

C. If $X \notin \mathrm{NP}$, then its complement $\bar{X} \notin$ co-NP

D. If P = PSPACE, then co-NP = NP

E. None of the above.

2 (10 pt)

We define a sequence of matrices $H_1, H_2, \ldots, H_k$ as follows.

1. $H_0 = [1]$

2. For $k > 0$, $H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$

Let v be a vector of length $n = 2^k$. Computing the product $H_k v$ in the usual way would take $O(n^2)$ arithmetic operations (additions, subtractions, multiplications). Describe an algorithm that can compute the product with fewer operations and write down the time complexity of your algorithm.

We can apply a divide and conquer algorithm:

Let $\cancel{V} = (V_1, V_2 \cdots V_{2^{k-1}}; V_{2^{k-1}+1} \cdots\cdots V_{2^k})^T$

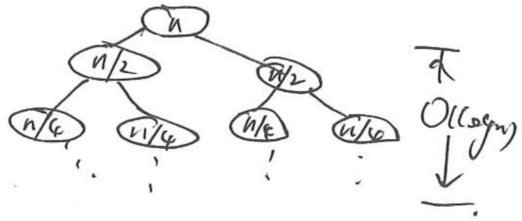
We set $V_U = (V_1, V_2 \cdots V_{2^{k-1}})^T$ and $V_D = (V_{2^{k-1}+1}, V_{2^{k-1}+2} \cdots V_{2^k})^T$

Notice that
$$H_k \cdot V = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix} \begin{pmatrix} V_U \\ V_D \end{pmatrix} = \begin{pmatrix} H_{k-1} V_U + H_{k-1} V_D \\ H_{k-1} V_U - H_{k-1} V_D \end{pmatrix}$$

And $H_{k-1} V_U$ and $H_{k-1} V_D$ is two sub-problems with size $\frac{n}{2}$ (2^{k-1})

the addition, substractions of $H_{k-1} V_U$ and $H_{k-1} V_D$ (~~between~~ $\frac{n}{2}$-sized vectors) take $O(n)$ time.

Hence we can write down the recursion equation.
$$T(n) = 2T(\tfrac{n}{2}) + O(n)$$



The depth of the recursion tree is $O(\log n)$

Hence the time complexity is $O(n\log n)$.

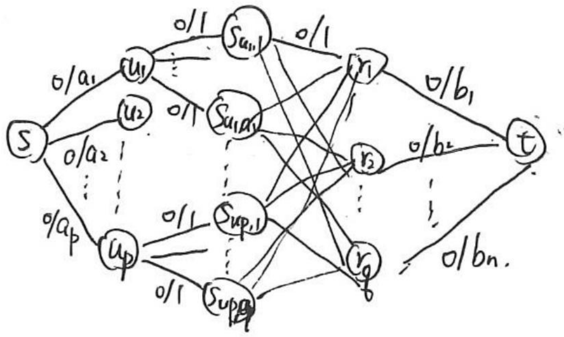
(or by the master theorm: $f(n) = O(n) = \Theta(n^{\log_2 2}) = \Theta(n)$,

So. $T(n) = O(n\log n)$).

3 (10 pt)

Students from several universities gather together for a social event. To increase social interaction, the event organizers want to assign these students to meeting rooms such that no two students from the same university are in the same room. Assume that there are p universities and there are a_i students from the i-th university. Also assume that there are q rooms and the j-th room can accommodate at most b_j students. Design a polynomial-time algorithm that outputs a valid room assignment, or returns an error message if no such assignment exists.

We can apply the ~~maxflow~~ network flow Algorithm, construct ~~the~~ the following graph. where (S) is the source, it connects to p universities (named $u_1 \cdots u_p$), then, each university node (u_i) connects to its students (named $S_{u_i,1}, S_{u_i,2} \cdots S_{u_i,a_i}$) then each student node connects to q room node (named $r_1 \cdots r_q$). then each room node connects to the sink (t):



The weights between university (u_i) and (S) is equal to the # of students from it (a_i)
~~from~~ weights between university node and student nodes are all 1
weights between. Students and rooms are also 1, and weights between room (r_i) and (t) ~~are~~ is equal to b_i.

Then run ford-fulkson algorithm on this graph to find a maxflow. then ~~each student~~ if the maxflow $= \sum_i a_i$, then there exist an arrangement. which each student will be uniqually assignedto a room. Becasue we set $W(u_i, S_{u_i,k}) = 1$. then we can know each student are assigned to which room. and output it. otherwise ~~the~~ indicates such assignment doesn't exist.

4 (10 pt)

Suppose a cashier needs to make change for v RMB, and has available bills of denominations $x_1, ..., x_k$, where $1 \leq x_1 < ... < x_k$ are positive integers. Give an algorithm to compute the number ways to make change, ignoring the order of the bills. For example, when $v = 11$ and there are 3 types of bills of denominations 1, 5 and 10, there are 4 ways to make change, namely $\{10, 1\}, \{5, 5, 1\}, \{5, 1, 1, 1, 1, 1, 1\}$ and $\{1, \ldots, 1\}$ with 11 1's.

What is the running time of your algorithm?

~~We use can apply a dynamic programming algorithm.~~

~~the base case is, if $v = x_1$, we have only one way to assign~~ it.

~~And if $v < x_1$, we have no ways to assign it. First~~

~~Consider the total amout V contains n_1 number of x_1, n_2 number of x_2 ... and # of x_k in n_k, Let $f(v)$ denotes # of assignments of value v, then~~ uses ~~$\{x_1, x_2 \ldots x_k\}$ $f(v, n)$~~

~~$f(v) = \max\{f(v-x_1), f(v$~~ ~~Let $X = \{x_1, x \ldots x_k\}$~~

~~$f(v) = \max\{f(v-x_1) x_2)$ then~~

Let $f(v, X)$ denotes the number of assignments use $X = \{x_1, x_2 \ldots x_k\}$ to make charge of value v

~~$f(v) = f(v)$~~

$$\boxed{\begin{aligned} f(v, X) &= f(v - x_1, X \setminus x_1) + f(v - 2x_1, X \setminus x_1) \cdots + f(v - \lfloor \tfrac{v}{x_1} \rfloor x_1, X \setminus x_1) \\ &+ f(v, X \setminus x_1). \end{aligned}}$$

all possible conditions are for use $0, 1, 2 \cdots \lfloor \frac{v}{x_1} \rfloor$ number of x_1 cashes.

$\Rightarrow$ Hence we get the recursion: $f(v, X) = \sum_{i=0}^{\lfloor \frac{v}{x_1} \rfloor} f(v - i x_1, X \setminus x_1)$

~~The~~ For example, $X = \{1, 5, 10\}$, and $v = 11$, we calculate: $f(11, \{5, 10\})$, $f(10, \{5, 10\})$ ... $f(1, \{5, 10\})$, $f(0, \{5, 10\})$, and sum them together.

The base case is, if $v = x_1$, then there is only one ~~case~~ assignment: just use one x_1 (for $x_1 < x_2 \cdots < x_k$, so we can use $x_2 \cdots x_k$).

and if $v < x_1$, there's no such legal assignment.

The time complexity is: ~~Because~~ for $|X| = k$, we ~~need~~ use $T(v, k)$ to denote the time complexity:

$$\begin{aligned} T(v, k) &= T(v - x_1, k-1) + \cdots + T(v - \lfloor \tfrac{v}{x_1} \rfloor x_1, k-1) \\ &\leq T(v-1, k-1) + \cdots + T(v-1, k-1) \\ &\leq v \cdot T(v-1, k-1). \end{aligned}$$

So, the time complexity is $T(v, k)$ ~~$= O(\frac{v!}{k!})$~~

$T(v, k) = O(v \, T(v-1, k-1)) = O(v(v-1) \, T(v-2, k-2)) \cdots = O(\frac{v!}{(v-k)!})$.

Or can be expressed as $T(n) = O(v^k)$, When k is fixed, it's polynomial to v.

5 (10 pt)

Recall from class that there is an efficient 3/2 approximation algorithm for the *metric* TSP problem, where for any three nodes i, j and k, we have $w_{i,j} + w_{j,k} \geq w_{i,k}$ ($w_{u,v}$ denotes the weight of the edge from node u to v). Now, consider the general TSP problem, where the edges are allowed to have *arbitrary* weights. Prove that for any constant $k \geq 1$, there is no polynomial time k-approximation algorithm for general TSP, unless $P = NP$.

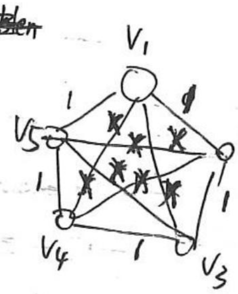
Hint: Recall that finding a Hamiltonian Cycle in a graph is NP-complete. Construct a graph which includes some edges with very large weights, such that finding an approximate TSP in this graph allows finding a Hamiltonian Cycle in another graph.

~~Consider the following construction of a TSP problem~~

Consider the following graph construction of TSP:

we have a graph $G(V,E)$. $V = \{V_1, V_2 \dots V_n\}$.

We set the weight between $e(V_i, V_{i+1})$ and $e(V_n, V_1)$ to be 1. and other edges $e(V_i, V_j)$ to be X, where X is ~~very~~ a very large number. ~~(arbitrary positive i~~. for example, the ~~fo~~ (fig.1) shows when there are 5 vertices.



(fig.1)

Then we proof ~~by contradiction~~ it: First, we know the optimal TSP in this graph is $L^* = n$. (Traverse $V_1 \to V_2 \dots \to V_n \to V_1$), and any other ~~path~~ TSP path has Length $L \geq X$ (at least contains one of other edges). Assume we have such a poly-time k-approximate algorithm. it can either find $L = L^*$ or $L \geq X$., for we can set X arbitray large, so WLOG, set $\underline{X > kL^* = kn}$. then ~~the~~ if the algorithm is correct, it can only output the optimal answer, for any other solution is incorrect. Then it must find the Hamiltonian circuit in this graph, But we know Hamiltonian Circuit problem is NP-complete, which indicates this approximate algorithm can't run in poly time if $P \neq NP$.

Hence, this contradiction shows unless $P = NP$. there's no poly-time TSP approximation algorithm (For we must find the Hamiltonian Cycle in some cases. which is NP-complete)

6 (10 pt)

Suppose we are given three $n \times n$ matrices A, B and C, and we want to check whether $AB = C$. One method is to directly multiply A and B and compare the result to C. This would take about $\Omega(n^{2.37})$ time using the fastest known matrix multiplication algorithm. However, since we only need to check whether $AB = C$, we can actually do better.

Give a Monte Carlo randomized algorithm which correctly decides whether $AB = C$ with $\Omega(1)$ probability, and runs in $O(n^2)$ time. Argue why your algorithm succeeds with $\Omega(1)$ probability and runs in $O(n^2)$ time.

Hint: Consider multiplying both sides of $AB = C$ by a vector. How much time does this take? What does the result tell you?

~~We know that~~ $\cancel{C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}}$ ~~(C_{ij} is the multiplication of i^{th} row vector in A and j^{th} column vector in B)~~

~~Cons~~ Consider an $n \times 1$ vector v. We know that.

~~if $ABv = Cv$, the~~ the i^{th} element of Mv (where M is $n \times n$ matrix) is $\sum_{k=1}^{n} M_{ik} \cdot v_k$. So if the i^{th} element of $A \cdot B \cdot v$ and $C \cdot v$ is not equal, then the i^{th} row of AB and C can't be the same.

And the mutiplacation of a $n \times n$ matrix (AB or C) and v is $O(n^2)$. So we use a random vector v ($n \times 1$) to multple AB and C. $\vec{v}$ is a random $n \times 1$ vector with real number

If the result of ABv and Cv are not equal, then AB must not equal to C. But if $AB \neq C$. ~~the~~ ABv may also equals to Cv, then we calculate the false probability.

The false case needs: for ~~all $i \in \{1,2\}$~~, all $i \in \{1,2,\cdots n\}$, $\sum_{k=1}^{n} (AB)_{ik} \cdot v_k = \sum_{k=1}^{n} C_{ik} \cdot v_k$

$\Rightarrow$ for all $i \in \{1, 2 \dots n\}$, $\sum_{i=1}^{n} (AB-C)_{ik} \cdot v_k = 0$. that means every row vector in $(AB-C)$ porpendicular to $\vec{v}$. ~~By the property of real number.~~

$\Rightarrow \cos\langle (AB-C)_i, \vec{v} \rangle = 0$, But $\cos\langle (AB-C)_i, \vec{v} \rangle \in [-1, 1]$. By the property of real number (实数的稠密性). There are infinite many $x \in [-1, 1]$. So, the probability of: $Pr(\sum_{i=1}^{n} (AB-C)_{ik} v_k = 0) \to 0$, So, ~~with~~ with probability $P = \Omega(1)$. we can infer that if $ABv = Cv$, then $AB = C$.

出错的概率~~[illegible]~~的测度为0