Practice exam

20. januar 2025 16:26

Example exam in AADS 2024–2025

Københavns Universitet

Instructions

You have four hours to complete the exam. You are allowed to use aids but you are not allowed to use your own computers, tablets, calculators, mobile phones or headphones.

The exam has two parts. A "multiple-choice part" and an "open part". There are 12 multiple-choice questions and 4 open questions. Each of the 12 multiple choice question is worth 6 points. Each of the open questions is worth 7 points. The total number of points is 100.

Multiple-choice part. You should start your submission with your solutions to the multiple-choice part. Your submission should start with 12 rows. And on row i you should write i and then your answers in numbers. As an example, it should start in this format:

1. 2, 3, 5
2. 2
3. 5, 6

⋮

12. 5

In all multiple-choice questions, there could be more than one true option. To get a full grade (6 points), you should mark exactly all the correct options. If you mark a non-empty subset of the correct options you get half the points (3 points). In any other case, you get 0 points.

Open part. The solutions to the open questions should appear in the order of the questions. The solutions to question i should start with the title "Solution to question i". Write your solutions clearly and accurately. Your grade will be determined both by the correctness and by the clarity of your answers.

Notation. Throughout the exam we use $[n]$ to denote the set $[n] = \{1, 2, \ldots, n\}$.

3

Ω = lower bound
O = upper Bound
Θ = Tight

Rand Min Cut (G)
min cut pr ≥ $\frac{2}{n(n-1)}$

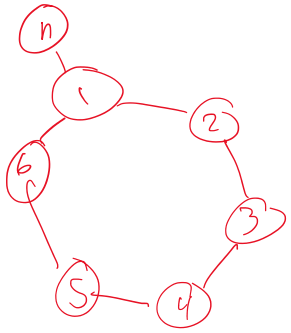
? 0

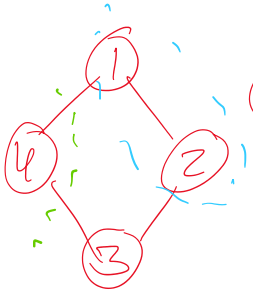
I. Multiple-choice questions

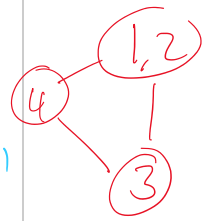
1 Random cuts (6%)

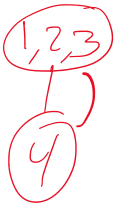
Let $n > 3$ be an integer and let G_n be the n-cycle graph with one edge extra attached to it. That is, its vertex set is $[n+1]$ and its edges are $\{1,2\},\{2,3\},\ldots,\{n-1,n\},\{n,1\}$ and $\{1,n+1\}$. Run the randomized min-cut algorithm for one iteration on G_n (that is, just until the algorithm finds a cut—this cut could be a non-min cut). What is the probability p_n that algorithm outputs a minimum cut?

1. $p_n \geq \Omega(\frac{1}{n})$.
2. $p_n \geq \Omega(\frac{1}{n^2})$. ✗
3. $p_n \geq \Omega(1)$.
4. $p_n \leq O(\frac{1}{n})$.
5. $p_n \leq O(\frac{1}{n^2})$.
6. $p_n \leq O(1)$.









1 Pr = $\frac{1}{n}$
2 pr = $\frac{1}{n-1}$
3 $\frac{1}{n-2}$
4 $\frac{1}{n-3}$
5 $\frac{1}{n-4}$
6

pr = $1 - \left(\sum_{i=0}^{n-1} \frac{1}{n-i}\right) = 1 - \frac{n-1}{}$

n − 0
n − 1
n − 2
n − 3
n − 4
11

$\left(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \frac{1}{n-3}\right)$ n times

$O(\frac{1}{n^2})$

$Pr(H(x) = a) = \frac{1}{n}$

$Pr(H(y) = a) = \frac{1}{n}$

2 Hashing (6%)

Let $n > 10$ be an integer. Let H be a strong universal hash function from $[n]$ to $[n]$. That is, $\Pr[H(x) = a, H(y) = b] = \frac{1}{n^2}$ for all $a, b \in [n]$ and all distinct $x, y \in [n]$. Let C be the number of collisions:

$$C = \{(x, y) \in [n] \times [n] : H(x) = H(y)\}.$$

(It is a random variable.) What is the expected value of C?

1. Exactly n.
2. Exactly $n - 1$.
3. At most $10n$.
4. At most $\frac{n}{2}$.
5. At most $\sqrt{n}$.

Col with a

Chances Both x and another hashes to $a = Pr(H(x)=a) \cdot \sum_{i=1}^{n-1} Pr(H(i)=a) = \frac{1}{n} \cdot \frac{n-1}{n} = \frac{n-1}{n^2}$

this we do n times giving us $\frac{n(n-1)}{n^2} = \frac{n-1}{n}$

this we do for n variables

$\frac{n-1}{n} \cdot n = n-1$

n Collisions when x=y

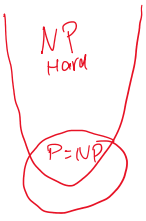
3

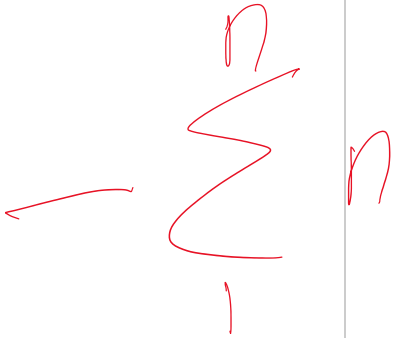
3 General complexity (6%)

Which of the following is true?

2
0

1. There is a function $f : \{0,1\}^* \to \{0,1\}$ that can not be computed by a Turing machine. infinite
2. There is an integer n and there is a function $f : \{0,1\}^n \to \{0,1\}$ so that f can not be computed by a Turing machine.
3. If P = NP, then SAT can be decided in time $O(2^{\sqrt{n}})$. SAT can be solved in $O(k^n)$
4. If P = NP, then there is an NP-hard language L so that L $\notin$ P.





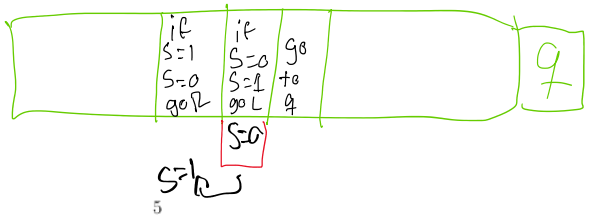
3

4 Small languages (6%)

We say that a language $L \subset \{0,1\}^*$ is small if for every integer $n > 1$, we have that $|L \cap \{0,1\}^n| \leq n^2$. Which of the following is true?

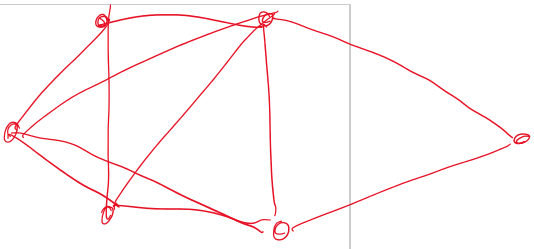
1. There is a small language L that can not be decided by a Turing machine.
2. There is a small language L that is not in P.
3. There is a small language L in P.
4. If L is small and $L \in$ P then the language $\{(1^n, k) : |L \cap \{0,1\}^n| = k\}$ can be decided by a Turing machine.
5. If L is not small, then L is not in P.
6. If L is small, then L is in P.

the accepted languages are at most
n long and there are $\leq n^2$ accepted languages
not quite true.
there are less than n^2 accepted languages
that are n long



$Y_1 = X_1 = 1$

$Z = \neg X_1 = 1$



5 Linear programming (6%)

Let

$$A = \begin{bmatrix} 2 & 1 \\ -5 & 1 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad b = (5, 5, 0, 6).$$

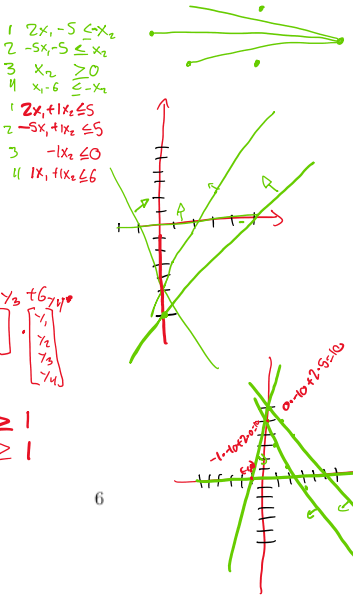
Let $P = \{x \in \mathbb{R}^2 : Ax \leq b\}$. Which of the following is true?

1. The polytope P has 4 vertices.
2. Let $F(x) = \langle c, x \rangle$ where $c = (1, 1)$. Then, $\max\{F(x) : x \in P\} = 5$.
3. Let $G(x) = \langle c, x \rangle$ where $c = (-10, 2)$. There is $x \in P$ so that for all $x' \in P$ so that $x' \neq x$ we have $G(x') < G(x)$.

2. Let $F(x) = \langle c, x \rangle$ where $c = (1, 1)$. Then, $\max\{F(x) : x \in P\} = 5$.

3. Let $G(x) = \langle c, x \rangle$ where $c = (-10, 2)$. There is $x \in P$ so that for all $x' \in P$ so that $x' \neq x$ we have $G(x') < G(x)$.

4. Let $Q = \{y \in \mathbb{R}^4 : A^T y \geq c\}$ where $c = (1, 1)$, and let $H : \mathbb{R}^4 \to \mathbb{R}$ be defined by $H(y) = \langle b, y \rangle$. Then, $\min\{H(y) : y \in Q\} \geq 6$.



Linear Programming (6%)

Let

$$A = \begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 0 & -2 \\ 1 & -1 \end{bmatrix}$$

and

$$b = (4, 6, -2, 3).$$

Let

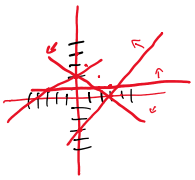
$$P = \{x \in \mathbb{R}^2 : Ax \leq b\}.$$

Which of the following is true?

1. The polytope P has 5 vertices.
2. Let $F(x) = \langle c, x \rangle$, where $c = (2, 1)$. Then, $\max\{F(x) : x \in P\} = 8$.
3. Let $G(x) = \langle c, x \rangle$, where $c = (-1, 4)$. There is $x \in P$ so that for all $x' \in P$ with $x' \neq x$, we have $G(x') < G(x)$.
4. Let

$$Q = \{y \in \mathbb{R}^4 : A^T y \geq c\},$$

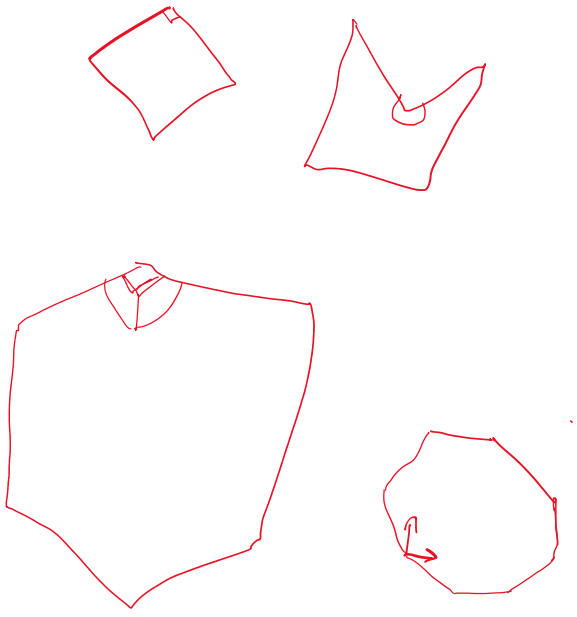
where $c = (1, 2)$, and let $H(y) = \langle b, y \rangle$. Then, $\min\{H(y) : y \in Q\} \geq 7$.



6 Art gallery (6%)

In this question, the guards can guard in a 90^o angle (if the guard is placed at the point $x \in \mathbb{R}^2$, there are two perpendicular lines through x not necessarily parallel to the axis and the guarded area is one of the quadrants defined by the lines). What is the minimum number k of 90^o-angle guards that are needed to guard a convex polygon $P \subset \mathbb{R}^2$?

1. $k = 1$
2. $k = 2$
3. $k = 3$
4. We need more information about P to decide k.



6

7 Helpers (6%)

We apply the algorithm MakeMonotone from page 53 of Chapter 3 of the book *Computational Geometry: Algorithms and Applications* to the polygon in Figure 1. What are the helpers of the edges bc, ef, hi just after the vertex e has been handled?

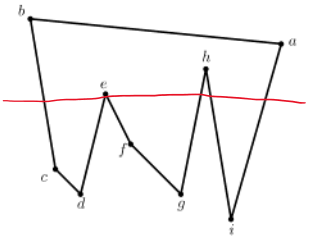


Figure 1: A polygon

1. ~~$helper(bc) = b, helper(ef) = e, helper(hi) = h$~~
2. ~~$helper(bc) = h, helper(ef) = h, helper(hi) = a$~~
3. ~~$helper(bc) = c, helper(ef) = f, helper(hi) = i$~~
4. $helper(bc) = e, helper(ef) = e, helper(hi) = h$

3

8 Approximate covers (6%)

The SET-COVER problem generalizes the VERTEX-COVER problem. Every input to VERTEX-COVER is also an input to SET-COVER. In class, we have seen a (matching-based) approximation algorithm APPROX-VERTEX-COVER for vertex cover and a greedy approximation algorithm GREEDY-SET-COVER for set cover.

Let $G = (V, E)$ be an input to VERTEX-COVER. Which of the following is true?

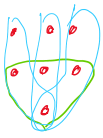
1. We can reduce the problem of finding a minimum vertex cover in G to an instance of set cover $I = (X, \mathcal{F})$ where X is E and there is one set in $\mathcal{F}$ for each $v \in V$.
2. We always get a better approximation ratio by using the algorithm GREEDY-SET-COVER on the instance G instead of using the approximation algorithm APPROX-VERTEX-COVER for vertex cover directly on G.
3. There are graphs on which using GREEDY-SET-COVER on the instance I performs worse than applying APPROX-VERTEX-COVER directly on G.
4. The algorithm APPROX-VERTEX-COVER does not necessarily find a matching in G of the maximum possible size.
5. The algorithm APPROX-VERTEX-COVER for vertex cover always finds a matching in G of the maximum possible size.

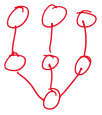
not true

? How

Not Max

no as there would be all vertex

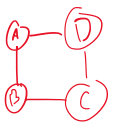


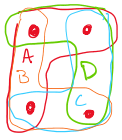


vertices

Family of subset (group of vertex)

$I = (X, F)$





9 FPTAS (6%)

For some optimization problem, we have a PTAS which achieves an approximation ratio of $1+\varepsilon$ in $f(n,\varepsilon)$ time for any fixed $\varepsilon > 0$. For which of the following cases is it an FPTAS?

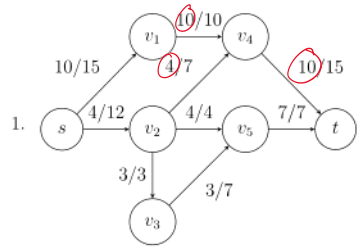
1. $f(n,\varepsilon) = n/\varepsilon$.
2. $f(n,\varepsilon) = (1/\varepsilon)^n$. not Pol
3. $f(n,\varepsilon) = (1/\varepsilon)^{\log n}$. not Pol
4. $f(n,\varepsilon) = (\log n)^{1/\varepsilon}$.
5. $f(n,\varepsilon) = 2^{\log n} + 2^{\log(1/\varepsilon)}$. not Pol
6. $f(n,\varepsilon) = 100^{\log n} \cdot 2^{100\log(1/\varepsilon)}$. not Pol

6

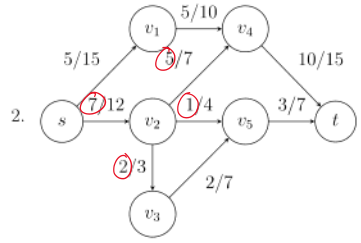
10 Valid flows (6%)

Which of the following examples of flows in a flow network are valid?

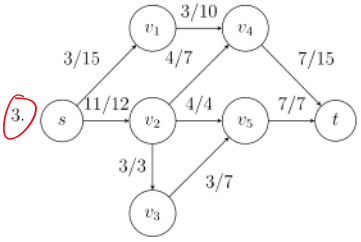
1.



2.

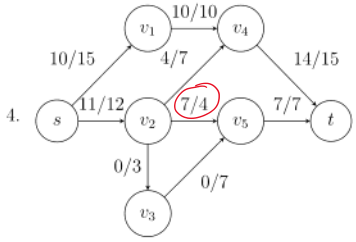


3.

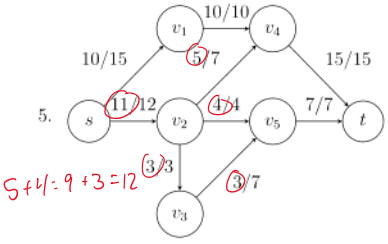


11 - 4 - 4 - 3

4.

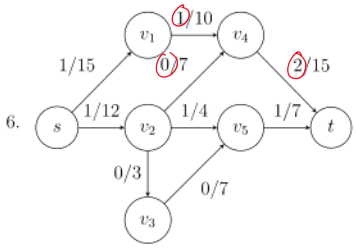

v_1
v_4
s
v_2
v_5
t
v_3
10/10
10/15
4/7
14/15
11/12
7/4
7/7
0/3
0/7

5.


v_1
v_4
s
v_2
v_5
t
v_3
10/10
10/15
5/7
15/15
11/12
4/4
7/7
3/3
3/7

5+4+9+3=12

6.


v_1
v_4
s
v_2
v_5
t
v_3
1/10
1/15
0/7
2/15
1/12
1/4
1/7
0/3
0/7

11 Max flow (6%)

In this course we defined the flow value of a valid flow f in $G = (V, E)$ from a source s to a sink t as the amount of flow going out of the source minus the amount of flow going into the source:

$$V_s(f) = \Big(\sum_{v \in V} f(s, v)\Big) - \Big(\sum_{v \in V} f(v, s)\Big)$$

Consider the case where we define it as the amount of flow going into the sink minus the amount of flow going out of the sink:

$$V_t(f) = \Big(\sum_{v \in V} f(v, t)\Big) - \Big(\sum_{v \in V} f(t, v)\Big).$$

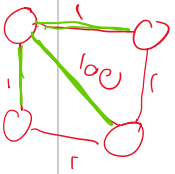
Are the two definitions the equivalent (that is, $V_s(f) = V_t(f)$) and why?

1. They are not equivalent because of the capacity constraint.
2. They are equivalent because of flow conservation.
3. They are not equivalent because of flow conservation.
4. They are equivalent because of the max-flow min-cut theorem.
5. They are equivalent because of the capacity constraint.
6. They are not equivalent because of the max-flow min-cut theorem.

3

12 TSP algorithm (6%)

We showed in class that there is a polynomial-time 2-approximation algorithm for the TSP problem when the triangle inequality holds. Which of the following statements is true for an instance (G, c) of TSP when the triangle inequality is *not* satisfied?



1. Let T be any spanning tree of G. Then the cost of T is always at most the cost of the shortest TSP tour H^*.
2. Let T be any minimum spanning tree of G. Then the cost of T is always at most the cost of the shortest TSP tour H^*.
3. It may be the case that every spanning tree of G has larger cost than the cost of H^*.
4. It may be the case that the cost of the full-walk W on T (as defined in the algorithm) has larger cost than twice the cost of T.
5. It may be the case that the cost of the tour H constructed by the algorithm from the full-walk W is larger than the cost of W.

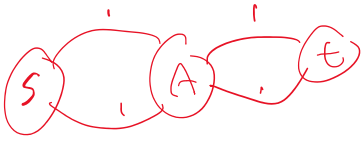
30

II. Open questions

1 Flows (7%)

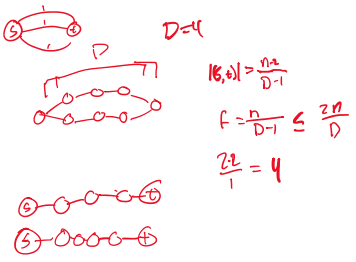
(a) Define the notion "augmenting path".

(b) Let G be a flow network with a source s and a sink t. Denote by n the number of vertices in G. Assume that the number of edges on the shortest path from s to t is D. Also assume that for every node $v \notin \{s, t\}$, there is a unique path from s to v. Assume that all capacities are in $[0, 1]$. Prove that the maximum flow value on G is at most $\frac{2n}{D}$.



a) given a flow network $G = (V, E)$ and a

c) given a flow network $G = (V, E)$ and a flow f, an augmenting path p is a simple path p from s to t in the residual network G_f



b) 

We can prove this by induction
we start by assuming that the edge $(s,t) \notin G$ as else this wouldn't be true
we therefore have basecase $n=1$
as we have the edges (s,v) and (v,t) with max capacity 1 we get that the flow of G_1 is $f_{G_1} \leq 1$ this is equal to $\frac{2n}{D} = \frac{2\cdot 1}{2} = 1$

Now we have our induction step so we assume it is true for $k=n$ so now we look at $k=n+1$
if we add either the vertex to another vertex or set the flow to 0 then it must still hold
so we look at the case where we add the vertex to the source and sink
from the case with n we had $f \leq \frac{2n}{D}$
as we have only made n larger it holds

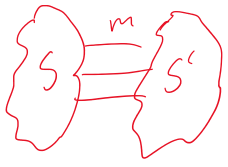
2 Randomized algorithms (7%)

(a) If we have a coin that lands on "head" with chance $p \in (0, 1)$, what is the expected number of time we need to toss it until we get the first "head"?

(b) Let $G = (V, E)$ be a graph with $|V| = 300$ vertices and $|E| = m$ edges. For $S \subseteq V$, denote by $e(S, S^c)$ the number of edges in G with one endpoint in S and one endpoint outside S. Prove that there is $S \subseteq V$ of size $|S| \leq 100$ so that $e(S, S^c) \geq \frac{4m}{9}$.

? $\frac{m}{2}$ $\frac{4m}{9} < \frac{m}{2}$

a) $Pr(\text{head}) = p$
expected amount of toss $= \frac{1}{p}$

b) 

if we for each edge place one endpoint in S and one in S'

16

3 Complexity (7%)

(a) Define the notion "the language L is NP-complete".

(b) Assume that P = NP. Show that there is a polynomial-time algorithm that takes as input a graph $G = (V, E)$, and outputs a set of vertices $K \subseteq G$ that forms a clique in G of maximum size.

a) the language L is NP-complete iff
1) $L \in NP$ and
2) $L' \leq_p L$ for every $L' \in NP$

this means that 1) we can verify it in polinomial time meaning

$L = \{x \in \{0,1\}^* \mid$ there is a $y \in \{0,1\}^*$ with $|y| = O(|x|^c)$ s.t. $A(x,y) = 1\}$

b) if P = NP then we must be able to solve VERTEX-COVER in polinomial time and since we know that CLIQUE $\leq_p$ VERTEX-COVER then we can also solve Clique problem in polinomial time

17

4 Approximation (7%)

(a) Consider a term of the form $\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4$ where $\ell_1, \ldots, \ell_4$ are literals. If the input to the term is chosen uniformly at random, what is the chance that the term evaluates to 1?

(b) If we run the RANDOM-ASSIGNMENT algorithm for MAX-4-SAT instead of MAX-3-SAT, is it still an $\frac{8}{7}$-approximation? If your answer is yes, prove it. If your answer is no, give a new approximation ratio and prove it.

a) we will first look at the chance that the term evaluates to 0 that means we must get $\neg l_1 \wedge \neg l_2 \wedge \neg l_3 \wedge \neg l_4$ the probabilities can be written as $Pr(\neg l_1 \wedge \neg l_2 \wedge \neg l_3 \wedge \neg l_4) = 0.5 \cdot 0.5 \cdot 0.5 \cdot 0.5 = 0.5^4 = \frac{1}{16}$

then we can take the opposite and we get $15/16$ chance of 1 ($\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$ $\frac{1}{16}$)

b) we follow the same thought pattern from max 3 sat

$Pr[\neg C_i] = Pr[\neg l_1] \cdot Pr[\neg l_2] \cdot Pr[\neg l_3] \cdot Pr[\neg l_4] = \left(\frac{1}{2}\right)^4 = 1/16$

$Pr[C_i] = 1 - Pr[\neg C_i] = 1 - 1/16 = 15/16$

we then have

$X_i = \sum_{i=1}^{n} [C_i] = \#$ sat clauses

$$E[X] = E\left[\sum_{i=1}^{n} [C_i]\right] = \sum_{i=1}^{n} E[C_i] = \sum_{i=1}^{n} \frac{15}{16} = \frac{15n}{16}$$

Approx Ratio $\frac{C^*}{C} = \frac{C^*}{15n/16} \leq \frac{n}{\frac{15n}{16}} = \frac{16}{15}$

So the approx Ratio is Better

---

c) Parameterized Complesite

① as we can solve Clique $\forall r^k$

1. VEB Tree Operations Complexity

Let u denote the universe size of a Van Emde Boas (VEB) tree. Which of the following statements about the time complexity of operations on a VEB tree is correct?

1. INSERT operation takes $O(\log \log u)$ time.
2. DELETE operation takes $O(\log u)$ time.
3. SUCCESSOR operation takes $O(\log \log u)$ time.
4. MEMBER operation takes $O(1)$ time.
5. MINIMUM operation takes $O(\log u)$ time.

2. Structure of VEB Trees

Which of the following attributes are part of a Van Emde Boas (VEB) tree with universe size $u > 2$?

1. An array A to store all keys.
2. Attributes $\min$ and $\max$ to store the minimum and maximum keys, respectively.
3. A pointer to a summary structure of size $\sqrt{u}$.
4. A cluster array of $\sqrt{u}$ pointers, each pointing to a VEB tree of size $\sqrt{u}$.
5. A linked list to track predecessor and successor relationships.

Open Question

3. Recursive Structure of VEB Trees

Explain how recursion is used in the design of Van Emde Boas (VEB) trees to achieve $O(\log \log u)$ time complexity for operations like SUCCESSOR and PREDECESSOR. Include the role of the summary structure and the cluster array in your explanation.

1. Parameterized Complexity and Fixed-Parameter Tractability

Which of the following is true about Fixed-Parameter Tractable (FPT) algorithms in parameterized complexity?

1. An FPT algorithm has a running time of $f(k) \cdot n^{g(k)}$, where $f(k)$ and $g(k)$ are non-decreasing functions. $f(x) \cdot n^c$
2. An FPT algorithm's running time is $f(k) \cdot n^c$, where $f(k)$ is a computable function of k, and c is a constant independent of k and n.
3. FPT algorithms can solve NP-complete problems in polynomial time for all k. for a fixed k
4. The FPT class is a subset of the XP class.

---

2. Exact Exponential Algorithms for TSP

Consider the dynamic programming algorithm for solving the Traveling Salesman Problem (TSP). Which of the following statements are true about its complexity?

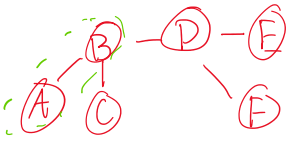
1. The algorithm runs in $O(n! \cdot n)$ time.
2. The algorithm uses $O^*(2^n)$ time.
3. The space complexity is $\Omega(2^n)$. computes $n^2 \cdot 2^n$ shortest paths
4. The algorithm's time complexity can be improved beyond $O^*(2^n)$ using dynamic programming techniques. would then prove P=NP?
5. The algorithm uses $O(n^2 \cdot 2^n)$ operations to compute the optimal solution. Paths

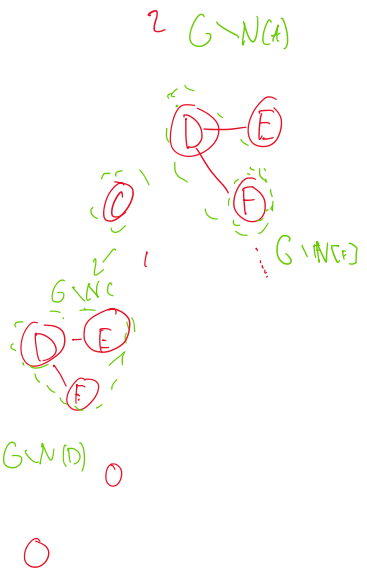
---

Open Question

3. Branching Algorithm for Maximum Independent Set (MIS)

Describe how the Branch & Reduce algorithm works for solving the Maximum Independent Set (MIS) problem. Provide an example of how the algorithm branches and discuss its time complexity. Include a brief explanation of the recurrence used to analyze the algorithm.





Randomized Quicksort

1. Let A be an array of size $n > 1$. When running the Randomized Quicksort algorithm, which of the following statements about the pivot selection is true?
   1. The pivot is always the first element of the array. Random
   2. The pivot is selected randomly with equal probability from all elements in the array.
   3. The pivot is chosen as the median of the array.
   4. The pivot is selected randomly, but only from the first half of the array.
2. For an array of size n, the expected runtime of Randomized Quicksort is:
   1. $O(n \log n)$.
   2. $O(n^2)$.
   3. $O(n + \log n)$.
   4. $O(\log n)$.
3. During the partitioning step in Randomized Quicksort, what is the expected number of comparisons performed on an array of size n?
   1. n.
   2. $n - 1$.
   3. $n + \log n$. $2n \ln(n)$ $O(n \log n)$ $2nH_n = 2n \sum_{k=1}^{n} \frac{1}{k}$

3. During the partitioning step in Randomized Quicksort, what is the expected number of comparisons performed on an array of size n?
   1. n.
   2. $n - 1$.
   3. $n + \log n$.
   4. $2n \ln(n)$.
   5. n^2.
4. Which of the following is true about the worst-case runtime of Randomized Quicksort?
   1. The worst-case runtime is $O(n \log n)$.
   2. The worst-case runtime is $O(n^2)$, but it happens with very low probability.
   3. The worst-case runtime is $O(n^2)$ and occurs with probability $1/2$.
   4. The worst-case runtime never occurs due to randomization.

Hashing

5. Let H be a universal hash function that maps $[n]$ to $[n]$. What is the expected number of collisions when inserting k distinct elements into the hash table?
   1. k^2/n.
   2. $k \cdot n$.
   3. n/k.
   4. k/n.
6. In a hash table using chaining for collision resolution, which of the following is true about the expected length of a chain when k keys are inserted into a table of size m?
   1. The expected chain length is k/m, assuming uniform hashing.
   2. The expected chain length is m/k, assuming uniform hashing.
   3. The expected chain length is $k \cdot m$, assuming uniform hashing.
   4. The expected chain length is always 1, regardless of k and m.
7. Let H be a strong universal hash function mapping $[n]$ to $[m]$. For $x, y \in [n]$, $x \neq y$, which of the following probabilities is true?
   1. $\Pr[H(x) = H(y)] = 1/m$.
   2. $\Pr[H(x) = H(y)] = 1/n$.
   3. $\Pr[H(x) = H(y)] = 1/(m^2)$.
   4. $\Pr[H(x) = H(y)] = 1/(n^2)$.
8. In open-address hashing with linear probing, which of the following is true?
   1. The load factor should always be less than 0.5 to avoid clustering.
   2. Quadratic probing is always better than linear probing.
   3. The expected time for search is $O(1/(1-\alpha))$, where α is the load factor.
   4. Hash collisions are avoided completely if the hash function is universal.

Expected number of collisions

A common way to count collisions in the context of hashing is by counting the number of *colliding pairs* (i, j) with $i < j$. If k distinct elements $x_1, x_2, \ldots, x_k$ are hashed, a collision occurs between the pair (x_i, x_j) if and only if $H(x_i) = H(x_j)$.

- **Step 1**: For each pair (i, j) with $1 \leq i < j \leq k$, let the indicator random variable

$$X_{ij} = \begin{cases} 1 & \text{if } H(x_i) = H(x_j), \\ 0 & \text{otherwise.} \end{cases}$$

- **Step 2**: The total number of collisions X among these k items is then

$$X = \sum_{1 \leq i < j \leq k} X_{ij}.$$

- **Step 3**: We want $\mathbb{E}[X]$, the expected value of X. By the linearity of expectation,

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{1 \leq i < j \leq k} X_{ij}\right] = \sum_{1 \leq i < j \leq k} \mathbb{E}[X_{ij}].$$

- **Step 4**: Because the hash family is universal,

$$\mathbb{E}[X_{ij}] = \Pr[H(x_i) = H(x_j)] = \frac{1}{n}.$$

  This is true for *each* pair (i, j).

- **Step 5**: There are $\binom{k}{2} = \frac{k(k-1)}{2}$ such pairs. Therefore,

$$\mathbb{E}[X] = \sum_{1 \leq i < j \leq k} \frac{1}{n} = \binom{k}{2} \cdot \frac{1}{n} = \frac{k(k-1)}{2n}.$$

Hence, the expected number of collisions when inserting k distinct elements under a universal hash function is

$$\boxed{\frac{k(k-1)}{2n}}.$$