辅学 lesson4

Zhejiang University, Advanced Data Structrue and Algorithm Analysis

赵一帆

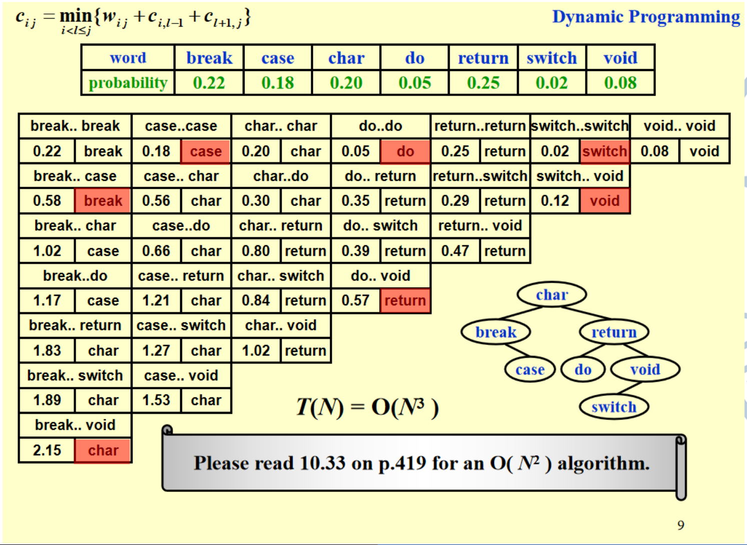
2025 年 12 月 14 日

目录

Dynamic programming

- What should we know?
- Coding.
- Analysis the table entry and recursive function.
- Classic problems.

Dynamic programming

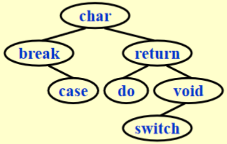


$$c_{ij} = \min_{i<l\le j}\{w_{ij} + c_{i,l-1} + c_{l+1,j}\}$$

Dynamic Programming

word	break	case	char	do	return	switch	void
probability	0.22	0.18	0.20	0.05	0.25	0.02	0.08

break.. break	case..case	char.. char	do..do	return..return	switch..switch	void.. void
0.22 break	0.18 case	0.20 char	0.05 do	0.25 return	0.02 switch	0.08 void
break.. case	case.. char	char..do	do.. return	return..switch	switch.. void	
0.58 break	0.56 char	0.30 char	0.35 return	0.29 return	0.12 void	
break.. char	case..do	char.. return	do.. switch	return.. void		
1.02 case	0.66 char	0.80 return	0.39 return	0.47 return		
break..do	case.. return	char.. switch	do.. void			
1.17 case	1.21 char	0.84 return	0.57 return			
break.. return	case.. switch	char.. void				
1.83 char	1.27 char	1.02 return				
break.. switch	case.. void					
1.89 char	1.53 char					
break.. void						
2.15 char						



$$T(N) = O(N^3)$$

Please read 10.33 on p.419 for an O(N^2) algorithm.

9

Dynamic programming

- How can we optimize the time complexity to $O(N^2)$?
- previous method
  - first loop, the number of different span length k is $O(N)$
  - second loop, the number of subproblems of fixed k is $O(N)$
  - recursive function, the time of updating one entry is $O(N)$
- Can we optimize one of them to $O(1)$?
- Actually we can combine the latter two steps, and optimize them to $O(N)$ as a whole.
- $root(i,j) \leq root(i,j+1)$(Donald Ervin Knuth, Turing Award in 1974)
- So is we only need to seek root of (i,j) between $(i,j-1)$ and $(i+1,j)$, which can complete the optimization if we view it layer by layer.(A layer indicates a fixed span length k)

Dynamic programming

- Other application of Dynamic Programming?
- If we want to transfrom a $H_1 \times W_1$ picture to a $H_1 \times W_2$, where $W_1 > W_2$
- If we only do sampling, the ratio changes, and distortion takes place.
- We can eliminate some pixels while holding other pixels unchanged.
- Which pixels should be deleted?
- How can we find optimal solution in polynomial time?

Greedy

- What should we know?
- The Activity Selection Problem, algorithm.
- Huffman Coding, algorithm, analysis.

Greedy

zgc & myc & cl Mid-term exam

If sorting can be done in $O(n)$ time, then we can implement the Huffman's algorithm so that it runs in $O(n)$ time.

Greedy

zgc & myc & cl Mid-term exam

If sorting can be done in $O(n)$ time, then we can implement the Huffman's algorithm so that it runs in $O(n)$ time.

If we can execute huffman code algorithm on a sorted array in $O(n)$, then we can select true.

Greedy

zgc & myc & cl Mid-term exam

If sorting can be done in $O(n)$ time, then we can implement the Huffman's algorithm so that it runs in $O(n)$ time.

If we can execute huffman code algorithm on a sorted array in $O(n)$, then we can select true. We can use "double queue" algorithm.

Greedy

zgc & myc & cl Mid-term exam

Given an optimal prefix code for n symbols, which of the following statements are true? (Recall that the average code length is the sum of the code length of all symbols, weighted by their frequencies.)

A.The maximum coding length of a symbol can be $n-1$

B.The minimum code length of a symbol can be 1

C.The average code length can be as large as $\Theta(n)$

D.The average code length can be as small as $O(1)$

Greedy

单选题

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. The coins of the lowest denomination（面额）is the cent.
(I) Suppose that the available coins are quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent). The greedy algorithm always yields an optimal solution.
(II) Suppose that the available coins are in the denominations that are powers of c, that is, the denominations are $c^0, c^1, \ldots, c^k$, for some integers c>1 and k>=1. The greedy algorithm always yields an optimal solution.
(III) Given any set of k different coin denominations which includes a penny (1 cent) so that there is a solution for every value of n, greedy algorithm always yields an optimal solution.
Which of the following is correct?
C.Statement (III) is false.

Greedy

单选题

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. The coins of the lowest denomination (面额) is the cent.
(I) Suppose that the available coins are quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent). The greedy algorithm always yields an optimal solution.
(II) Suppose that the available coins are in the denominations that are powers of c, that is, the denominations are $c^0, c^1, \ldots, c^k$, for some integers c>1 and k>=1. The greedy algorithm always yields an optimal solution.
(III) Given any set of k different coin denominations which includes a penny (1 cent) so that there is a solution for every value of n, greedy algorithm always yields an optimal solution.
Which of the following is correct?
C.Statement (III) is false.

Notice that we can find a counterexample in $(c_3 + 1, c_m + c_{m-1})$

Greedy

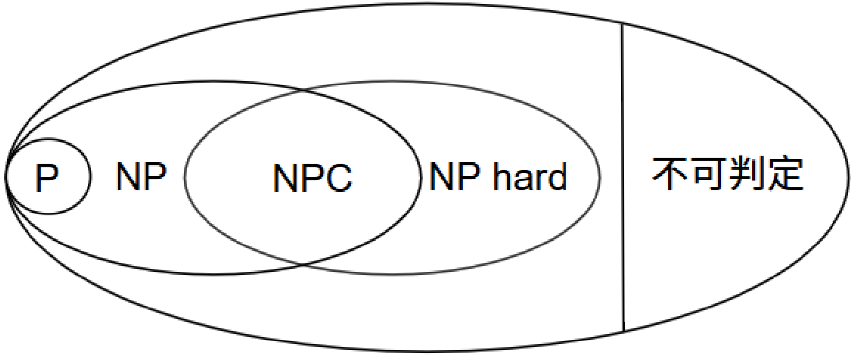
单选题

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. The coins of the lowest denomination（面额）is the cent.
(I) Suppose that the available coins are quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent). The greedy algorithm always yields an optimal solution.
(II) Suppose that the available coins are in the denominations that are powers of c, that is, the denominations are $c^0, c^1, \ldots, c^k$, for some integers c>1 and k>=1. The greedy algorithm always yields an optimal solution.
(III) Given any set of k different coin denominations which includes a penny (1 cent) so that there is a solution for every value of n, greedy algorithm always yields an optimal solution.
Which of the following is correct?
C.Statement (III) is false.

Notice that we can find a counterexample in (c_3+1, c_m+c_{m-1}) we can divide 25, 10 and 5 by 5 at the same time and we get a new problem 1, 2, 5, in which $c_3+1=6$ and $c_m+c_{m-1}=7$, so there is no counterexample.

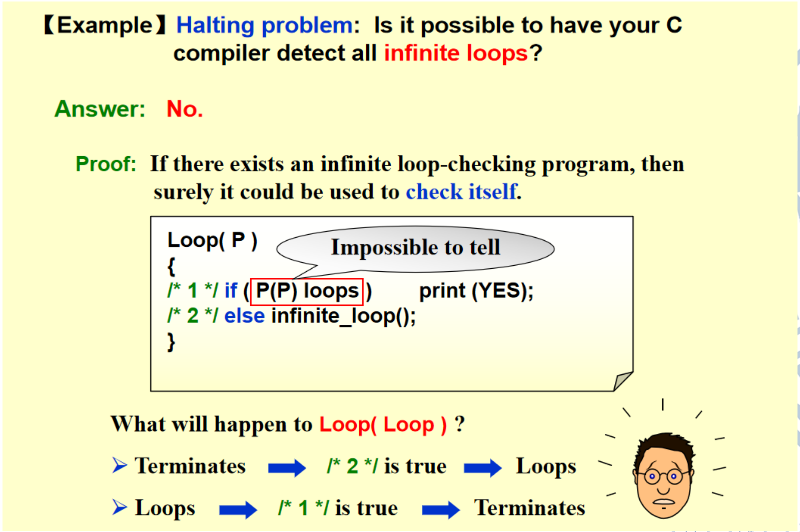
P and NP

What should we know?



decision problem.

P and NP



【Example】Halting problem: Is it possible to have your C compiler detect all infinite loops?

Answer: No.

Proof: If there exists an infinite loop-checking program, then surely it could be used to check itself.

```
Loop( P )
{
/* 1 */ if ( P(P) loops )        print (YES);
/* 2 */ else infinite_loop();
}
```

Impossible to tell

What will happen to Loop(Loop) ?

- Terminates ⟶ /* 2 */ is true ⟶ Loops
- Loops ⟶ /* 1 */ is true ⟶ Terminates

P and NP

- Halting Problem.
- When Turing Machine get a input(string), it must go into one of three situations: loop, halt and accept, halt and reject.
- Every Turing Machine can de encoded as a string.
- We assume there is a Turing Machine T_1 can decide whether a Turing Machine M getting itself as input can halt
  - If M halts and accects itself or halts and rejects itself, T_1 halts and accepts M
  - If M loops, T_1 halts and rejects M
- Then we can construct a Turing Machine T_2, when it get a input M
  - Run T_1 on M, if T_1 halts and accepts M, then T_2 loops forever.
  - If T_1 halts and rejects M, then T_2 halts and accepts M
- What if T_2 get itself as input?

P and NP

判断题

All $\mathcal{NP}$-complete problems are in $\mathcal{NP}$.
All $\mathcal{NP}$-complete problems are $\mathcal{NP}$-hard.
A problem is $\mathcal{NP}$-complete if and only if it is in $\mathcal{NP}$ and there is no polynomial time algorithms for it.
If a Probelm X is in $\mathcal{P}$, then the problem $\overline{X}$ is also in $\mathcal{P}$.
Let X be a problem in $\mathcal{NP}$. We know that both yes-instances (instances for which the answer is yes) and no-instances of X must have polynomial-length certificates.

P and NP

单选题

Suppose Q is a problem in $\mathcal{NP}$, but not necessarily $\mathcal{NP}$-complete. Which of the following is FALSE?

A.A polynomial-time algorithm for SAT would sufficiently imply a polynomial-time algorithm for Q.

B.A polynomial-time algorithm for Q would sufficiently imply a polynomial-time algorithm for SAT.

C.If $Q \notin P$, then $P \neq NP$.

D.If Q is $\mathcal{NP}$-hard, then Q is $\mathcal{NP}$-complete.

P and NP

单选题

Suppose Q is a problem in $\mathcal{NP}$, but not necessarily $\mathcal{NP}$-complete. Which of the following is FALSE?

A.A polynomial-time algorithm for SAT would sufficiently imply a polynomial-time algorithm for Q.

B.A polynomial-time algorithm for Q would sufficiently imply a polynomial-time algorithm for SAT.

C.If $Q \notin P$, then $P \neq NP$.

D.If Q is $\mathcal{NP}$-hard, then Q is $\mathcal{NP}$-complete.

It is generally believed that there are decision $NP \backslash P$ problems not $NP - Complete$. The graph isomorphism problem which have a $2^{O((logn)^c)}$ algorithm is a probable example.

P and NP

多选题

Which of the following statements is/are correct?

A.If some $\mathcal{NP}$-complete problem can be solved in polynomial time on deterministic Turing machine, then $\mathcal{P} = \mathcal{NP}$.

B.If a problem A can be reduced to problem B in polynomial time, then problem B can also be reduced to problem A in polynomial time.

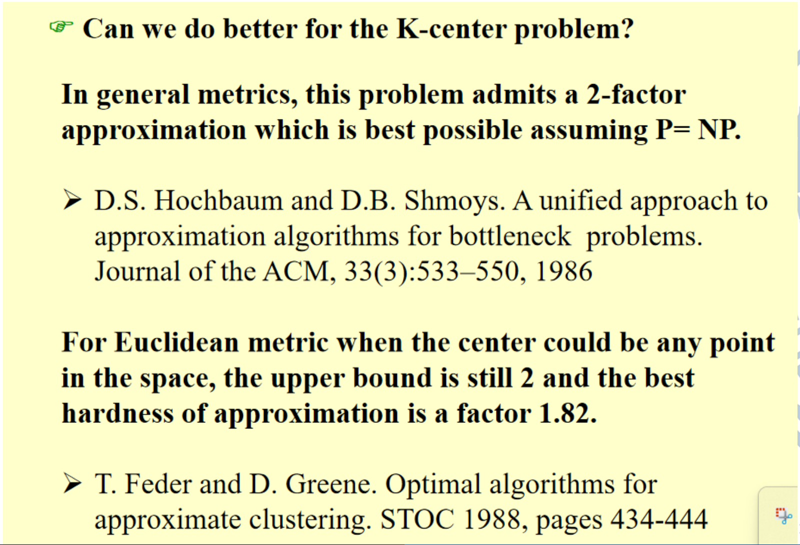
C.Given enough computational resources, such as time and space, any problem can be solved by a computer.

D.None of the other options is correct.

Approximation Algorithm

- What should we know?
- Bin packing: NF(2), FF(1.7), BF(1.7)
- Knapsack: 2-approximation, FPTAS
- K-center problem: 2-approx, two different versions(in metric graph and in Euclidean space).
  - If distances are unbounded, no approximation algorithm exists.
  - If we can only select given nodes in metric graph, 2-approximation algorithm can not be improved.
  - If we can select any nodes in 2-dimension Euclidean space, 2-approximation algorithm is known and no 1.82-approximation algorithm exists.
- TSP tour: 2-approximation, 1.5-approximation
- Online approximation algorithm
  - Online bin packing problem can not be α-approximation for any $\alpha < \frac{3}{5}$ unless $P \neq NP$

Approximation Algorithm



☞ **Can we do better for the K-center problem?**

In general metrics, this problem admits a 2-factor approximation which is best possible assuming P= NP.

➢ D.S. Hochbaum and D.B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. Journal of the ACM, 33(3):533–550, 1986

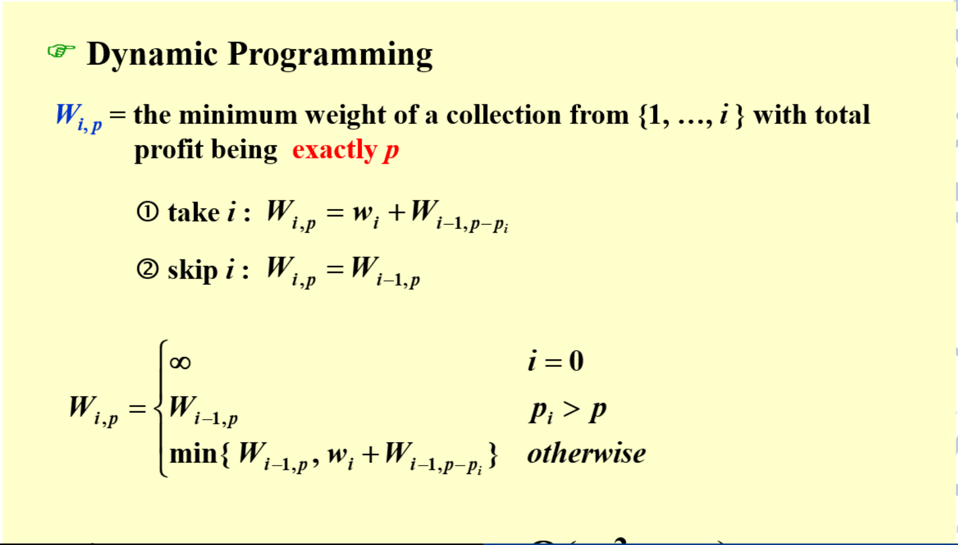
For Euclidean metric when the center could be any point in the space, the upper bound is still 2 and the best hardness of approximation is a factor 1.82.

➢ T. Feder and D. Greene. Optimal algorithms for approximate clustering. STOC 1988, pages 434-444

Approximation Algorithm

- PTAS, EPTAS and FPTAS?
- PTAS(Polynomial Time Approximation Scheme) is polynomial of n if ϵ is given, may be in the form as $n^{\frac{1}{\epsilon}}$
- EPTAS(Efficient Polynomial Time Approximation Scheme) is polynomial of n multiplies function of ϵ.
- FPTAS(Fully Polynomial Time Approximation Scheme) is polynomial of n multiplies polynomial of ϵ.

Approximation Algorithm



Dynamic Programming

$W_{i,p}$ = the minimum weight of a collection from {1, …, i} with total profit being **exactly p**

① take i : $W_{i,p} = w_i + W_{i-1,p-p_i}$

② skip i : $W_{i,p} = W_{i-1,p}$

$$W_{i,p} = \begin{cases} \infty & i = 0 \\ W_{i-1,p} & p_i > p \\ \min\{ W_{i-1,p}, w_i + W_{i-1,p-p_i} \} & otherwise \end{cases}$$

Approximation Algorithm

- Assume the rounded instance is $I' = \{p'_i\}$
- $solution(I) = \sum_{i \in S} p_i \geq k \sum_{i \in S} p'_i \geq k \sum_{i \in S*} p'_i \geq \sum_{i \in S*} p_i - k|S^*| \geq optimal(I) - kn$
- We only need to restrict $kn \leq \epsilon \times optimal(I)$, so $k = \lfloor \frac{\epsilon \times p_{max}}{n} \rfloor$, then the running time complexity is $O(\frac{n^3}{\epsilon})$, indicating it is a FPTAS.

Approximation Algorithm

判断题

For any instance, a 2-approximation algorithm must give a solution better than a 3-approximation algorithm.

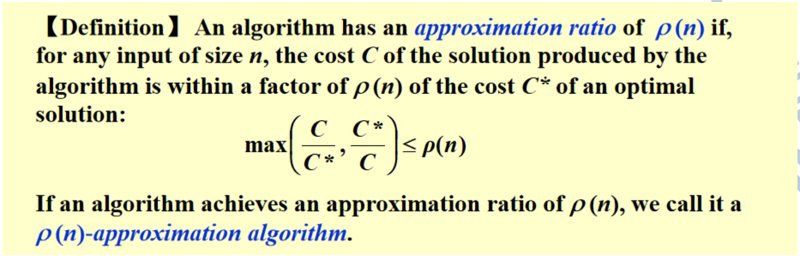
As we know there is a 2-approximation algorithm for the Vertex Cover problem. Then we must be able to obtain a 2-approximation algorithm for the Clique problem, since the Clique problem can be polynomially reduced to the Vertex Cover problem.

Approximation Algorithm

判断题

For any instance, a 2-approximation algorithm must give a solution better than a 3-approximation algorithm.

As we know there is a 2-approximation algorithm for the Vertex Cover problem. Then we must be able to obtain a 2-approximation algorithm for the Clique problem, since the Clique problem can be polynomially reduced to the Vertex Cover problem.


【Definition】 An algorithm has an *approximation ratio* of $\rho(n)$ if, for any input of size n, the cost C of the solution produced by the algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n)$$

If an algorithm achieves an approximation ratio of $\rho(n)$, we call it a *$\rho(n)$-approximation algorithm*.

Approximation Algorithm

单选题

Recall that Next Fit (NF) and First Fit (FF) are two simple on-line approximation algorithms for the Bin-Packing Problem, whose (asymptotic) approximation ratios are 2 and 1.7, respectively. Now we focus on a special class $I2$ of instances in which only two distinct item sizes appear. Which of the following statements is true by applying NF and FF on $I2$?SAT.

A.NF and FF both have improved approximation ratios.

B.NF has an improved approximation ratio, while FF does not.

C.FF has an improved approximation ratio, while NF does not.

D.Neither of NF or FF has an improved approximation ratio.

Approximation Algorithm

单选题

Recall that Next Fit (NF) and First Fit (FF) are two simple on-line approximation algorithms for the Bin-Packing Problem, whose (asymptotic) approximation ratios are 2 and 1.7, respectively. Now we focus on a special class $I2$ of instances in which only two distinct item sizes appear. Which of the following statements is true by applying NF and FF on $I2$?SAT.

A.NF and FF both have improved approximation ratios.

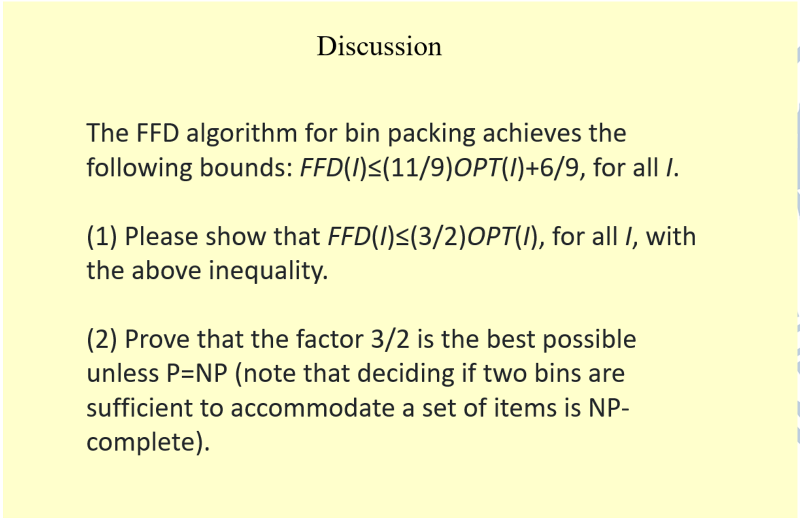
B.NF has an improved approximation ratio, while FF does not.

C.FF has an improved approximation ratio, while NF does not.

D.Neither of NF or FF has an improved approximation ratio.

Asymptotic a-approx means $algorithm(I) \leq a \times optimal(I) + b$ for any instance I Discussion on classification

Approximation Algorithm



Discussion

The FFD algorithm for bin packing achieves the following bounds: *FFD*(*I*)≤(11/9)*OPT*(*I*)+6/9, for all *I*.

(1) Please show that *FFD*(*I*)≤(3/2)*OPT*(*I*), for all *I*, with the above inequality.

(2) Prove that the factor 3/2 is the best possible unless P=NP (note that deciding if two bins are sufficient to accommodate a set of items is NP-complete).

Approximation Algorithm

- If two items are both bigger than $\frac{1}{2}$, then the First Fit can get the optimal solution.
- If two items are both smaller than or equal to $\frac{1}{2}$, then we leave out the bin which is the first bin less than $\frac{2}{3}$. According to the selection rule, these bins before this bin are all more than $\frac{2}{3}$, and the items after this bin are all bigger than $\frac{1}{3}$, so there are exactly two items in these bins after this bin except the last one.
  In another word, when we leave out the threshold bin and the last bin, other bins are all bigger than $\frac{2}{3}$.
- If one item in bigger than $\frac{1}{2}$ and another is smaller or equal to $\frac{1}{2}$
  - If the sum of them is bigger than 1, then the First Fit can get the optimal solution.
  - If the sum of them is smaller than or equal to 1, than we can also find the bin which is the first bin less than $1 - s$ which s is the size of the smaller item.

Approximation Algorithm

- You can also use threshold $\frac{1}{3}$ and $\frac{1}{2}$ to divide instances into 6 situations.
- If you want to drop out the word "asymptotic", you should analysis the special situations.
- Actually there is a tight instance of the 1.5-approx: consecutive n $\frac{1}{2} - \epsilon$ firstly and then consecutive n $\frac{1}{2} + \epsilon$, the optimal solution is n but we only get a solution $\frac{3}{2}$ n using First Fit.

Approximation Algorithm

单选题

K-center problem: Given N cities with specified distances, one wants to build K warehouses in different cities and minimize the maximum distance of a city to a warehouse.
Which of the following is false? A.Given any constant >1, unless P $=$ NP, otherwise the K-center problem cannot be approximated within the factor if the graph G admits an arbitrary distance function.
B.If the graph G obeys metric distance, then there is a 2-approximation algorithm for the K-center problem.
C.The K-center problem can be solved optimally in polynomial time if K is a given constant.
D.If the graph G obeys Euclidean distance, then there exists a PTAS for the K-center problem.

Approximation Algorithm

单选题

Assume that you are a real world Chinese postman, which have learned an awesome course "Advanced Data Structures and Algorithm Analysis" (ADS). Given a 2-dimensional map indicating N positions $p_i(x_i, y_i))$ of your post office and all the addresses you must visit, you'd like to find a shortest path starting and finishing both at your post office, and visit all the addresses at least once in the circuit. Fortunately, you have a magic item "Bamboo copter $\&$ Hopter" from "Doraemon", which makes sure that you can fly between two positions using the directed distance (or displacement).
However, reviewing the knowledge in the ADS course, it is an NPC problem! Wasting too much time in finding the shortest path is unwise, so you decide to design a 2$-$approximation algorithm as follows, to achieve an acceptable solution.

Approximation Algorithm

单选题

Compute a minimum spanning tree T connecting all the addresses.
Regard the post office as the root of T.
Start at the post office.
Visit the addresses in order of a ( ) of T.
Finish at the post office.
There are several methods of traversal which can be filled in the blank of the above algorithm.
Assume that $P \neq NP$, how many methods of traversal listed below can fulfill the requirement?
* Level-Order Traversal
* Pre-Order Traversal
* Post-Order Traversal
A.0 B.1 **C.2** D.3

Approximation Problem

多选题

Let $G = (V, E)$ be a undirected graph with non-negative edge weights. We assume that all edge weights are distinct. Let T_G be the maximum spanning tree of G. For each $v \in V$, define $\max(v)$ be the maximum-weight edge incident to v. Let $S_G = \{\max(v) : v \in V)\}$. ($S_G$ can be extened to a spanning tree of G in linear time. Can you see how to do this?) Which of the following statements are true?

A.$S_G = T_G$.
B.$S_G \neq T_G$.
C.$S_G \subseteq T_G$**.**
D.$w(S_G) \geq w(T_G)/2$**.**

Approximation Problem

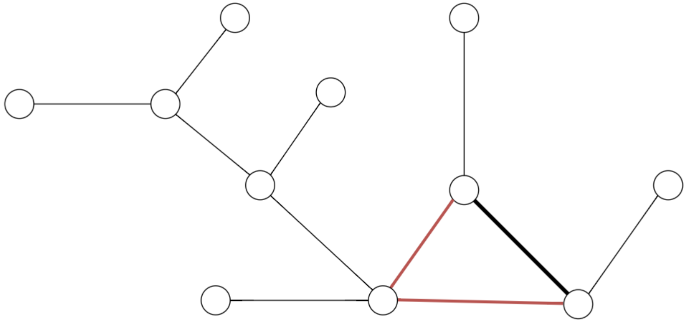
多选题

Let $G = (V, E)$ be a undirected graph with non-negative edge weights. We assume that all edge weights are distinct. Let T_G be the maximum spanning tree of G. For each $v \in V$, define $\max(v)$ be the maximum-weight edge incident to v. Let $S_G = \{\max(v) : v \in V)\}$. ($S_G$ can be extened to a spanning tree of G in linear time. Can you see how to do this?) Which of the following statements are true?

A.$S_G = T_G$.
B.$S_G \neq T_G$.
C.$S_G \subseteq T_G$.
D.$w(S_G) \geq w(T_G)/2$.

We can view the method as an approximzation algorithm to solve this problem: find a maximum weighted edge set which has no loops.

Approximation Problem



We use Kruskal algorithm to solve the MST(Maximum Spanning Tree) problem.
If the black edge is in S_G, then the only possible reason why it is excluded from T_G is these two red edges have both been select before the black one, indicating they are both have bigger weight than the black one, there is a contradictory. Thus $S_G \subseteq T_G$.