UniCODE 풀이

Solution Slide for UniCODE

김현수 박서영 박원 신승원 안윤표 이서윤 이종서

UNIST HeXA

2019년 11월 2일

Sponsor Companies

- Many companies funded us for holding Uni-CODE 2019.
- We will introduce such companies.

Sponsor - Startlink

STARTLINK

BAEKJOON

ONLINE JUDGE

Sponsor - Samsung Software Membership

SAMSUNG
SOFTWARE
MEMBERSHIP

Sponsor - NAVER

NAVER

Problems

#	Problem	Time Limit	Memory Limit
A	Command	500ms	256MB
B	Baba is Rabbit	1000ms	512MB
C	Fibonacci song	1000ms	512MB
D	What does UNIST stand for?	1000ms	512MB
E	Bus Route	1000ms	512MB
F	Clock	1000ms	512MB
G	Ctrl_cv	2000ms	512MB
H	Course	1000ms	512MB

Command

- # Submissions: 37(Onsite)
- # Accepted: 12(Onsite)
- First Solver: 한준구(Onsite), pichulia(Open)
- Proposed by: Won Park

Command

- Check whether the length of the given string is 7 or not.

Command

- Check whether the length of the given string is 7 or not.
- Check whether the given string is in the form of “AABAABB” or not.

Command

- Implementation in C/C++

```
printf("%d\n", strlen(a) == 7 &&
    a[0] == a[1] && a[0] == a[4] &&
    a[2] == a[3] && a[2] == a[5] && a[2] == a[6] &&
    a[0] != a[2]);
```

Baba is Rabbit

- # Submissions: 18(Onsite)
- # Accepted: 3(Onsite)
- First Solver: 한동규(Onsite), pichulia(Open)
- Proposed by: Jongseo Lee

Baba is Rabbit

- Use `std::map`(C++) or `dict`(Python) to assign an integer to each object and construct a graph.

Baba is Rabbit

- Use `std::map`(C++) or `dict`(Python) to assign an integer to each object and construct a graph.
- Then, the graph is acyclic.

Baba is Rabbit

- Use `std::map`(C++) or `dict`(Python) to assign an integer to each object and construct a graph.
- Then, the graph is acyclic.
- Use DFS/BFS to find objects that can be obtained from Baba's transformation. And sort such objects.

Baba is Rabbit

- Use `std::map`(C++) or `dict`(Python) to assign an integer to each object and construct a graph.
- Then, the graph is acyclic.
- Use DFS/BFS to find objects that can be obtained from Baba's transformation. And sort such objects.
- Time complexity is $O(N \log N)$.

Fibonacci song

- # Submissions: 32(Onsite)
- # Accepted: 0(Onsite)
- First Solver: -(Onsite), pichulia(Open)
- Proposed by: Jongseo Lee and Won Park

Fibonacci song

- First, we consider $\{f_n \mod M\}$.

Fibonacci song

- First, we consider $\{f_n \mod M\}$.
- One can easily observe that $\{f_n \mod M\}$ is periodic.

Fibonacci song

- First, we consider $\{f_n \mod M\}$.
- One can easily observe that $\{f_n \mod M\}$ is periodic.

Theorem

$\{f_n \mod M\}$ has period at most M^2.

Fibonacci song

- First, we consider $\{f_n \mod M\}$.
- One can easily observe that $\{f_n \mod M\}$ is periodic.

Theorem

$\{f_n \mod M\}$ has period at most M^2.

- Proof: Trivial from pigeon-hole principle.

Fibonacci song

- Therefore, the new sequence has period at most $4M^2$.

Fibonacci song

- Therefore, the new sequence has period at most $4M^2$.
- So we can pre-compute the one period of the new sequence in $O(M^2)$.

Fibonacci song

- Therefore, the new sequence has period at most $4M^2$.
- So we can pre-compute the one period of the new sequence in $O(M^2)$.
- Then, in $O(1)$ time, we can answer each query. Whole time complexity is $O(M^2 + Q)$.

Fibonacci song

- Therefore, the new sequence has period at most $4M^2$.
- So we can pre-compute the one period of the new sequence in $O(M^2)$.
- Then, in $O(1)$ time, we can answer each query. Whole time complexity is $O(M^2 + Q)$.
- Warning: Since input value is very large, one should use 64-bit integer type, such as `int64_t` or `long long`.

What does UNIST stand for?

- # Submissions: 0(Onsite)
- # Accepted: 0(Onsite)
- First Solver: -(Onsite), pichulia(Open)
- Proposed by: Jongseo Lee

What does UNIST stand for?

- How can we solve the problem if $len(W_1) = len(W_2) = \cdots = len(W_N) = 1$?

What does UNIST stand for?

- How can we solve the problem if $len(W_1) = len(W_2) = \cdots = len(W_N) = 1$?
- We can approach using dynamic programming, by defining $D[i][j]$ as the number of way $P_1 + P_2 + \cdots + P_i$ being prefix of "UNIST" of length j, where $i \leq N, j \leq 5$.

What does UNIST stand for?

- How can we solve the problem if $len(W_1) = len(W_2) = \cdots = len(W_N) = 1$?
- We can approach using dynamic programming, by defining $D[i][j]$ as the number of way $P_1 + P_2 + \cdots + P_i$ being prefix of "UNIST" of length j, where $i \leq N, j \leq 5$.
- Then, we have following recurrence relation.

What does UNIST stand for?

- How can we solve the problem if $len(W_1) = len(W_2) = \cdots = len(W_N) = 1$?
- We can approach using dynamic programming, by defining $D[i][j]$ as the number of way $P_1 + P_2 + \cdots + P_i$ being prefix of "UNIST" of length j, where $i \leq N, j \leq 5$.
- Then, we have following recurrence relation.

$$D[i][j] = D[i-1][j] + \begin{cases} D[i][j-1] & \text{if } W_i[0] == \text{"}UNIST\text{"}[j] \\ 0 & \text{otherwise} \end{cases}$$

What does UNIST stand for?

- Then, in general, we can generalize previous recurrence relation to solve the problem.

What does UNIST stand for?

- Then, in general, we can generalize previous recurrence relation to solve the problem.
  - with same definition of $D[i][j]$.

What does UNIST stand for?

- Then, in general, we can generalize previous recurrence relation to solve the problem.
  - with same definition of $D[i][j]$.
- Since the recurrence relation is hard to write, I omitted in this slide. However, it is still easy to implement so don't worry.

What does UNIST stand for?

- Then, in general, we can generalize previous recurrence relation to solve the problem.
  - with same definition of $D[i][j]$.
- Since the recurrence relation is hard to write, I omitted in this slide. However, it is still easy to implement so don't worry.
- Total time complexity is $O(N)$, with large constant factor.

Bus Route

- # Submissions: 8(Onsite)
- # Accepted: 2(Onsite)
- First Solver: 한준구(Onsite), yongjun18(Open)
- Proposed by: Yunpyo An

Bus Route

- Is this DFS, or BFS problem?

Bus Route

- Is this DFS, or BFS problem?
- You can solve this DFS or BFS. But, it have more easy way.

Bus Route

Theorem

Let $P(G)$ is the number of vertices at tree graph G such that $deg(v) = 1$. The least number of bus route is $\lceil P(G)/2 \rceil$

Proof by mathematical induction. Continue on next slide.

Bus Route

Theorem

Let $P(G)$ is the number of vertices at tree graph G such that $deg(v) = 1$. The least number of bus route is $\lceil P(G)/2 \rceil$

- Base step : $G(\{v, v'\}, vv')$ is easy to check.

Bus Route

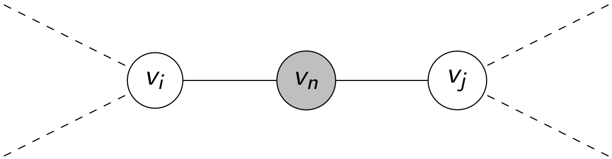
Theorem

Let $P(G)$ is the number of vertices at tree graph G such that $deg(v) = 1$. The least number of bus route is $\lceil P(G)/2 \rceil$

- Base step : $G(\{v, v'\}, vv')$ is easy to check.
- Induction hypothesis : $\lceil P(G)/2 \rceil$ is least number of bus route.

Bus Route

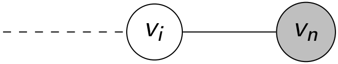
- Case 1



Add stop(vertex) between current nodes which degree of stops are over 1. $P(G) = P(G + v)$. And change the bus route which pass over road $v_i v_j$ to $v_i v_n v_j$. The number of minimum bus route is the same as G. So, $\lceil P(G)/2 \rceil = \lceil P(G + v)/2 \rceil$.

Bus Route

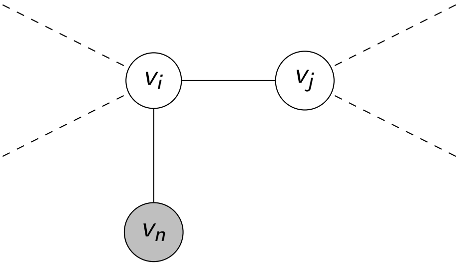
- Case 2



Add stop current node which degree of stop is 1. And change the bus route which pass over road v_i to $v_i v_n$. The number of minimum bus route is the same as G. So, $\lceil P(G)/2 \rceil = \lceil P(G+v)/2 \rceil$.

Bus Route

- Case 3



Add a new vertex next to the vertex that is not of degree 1.

Continue on next slide.

Bus Route

- $P(G+v) = P(G) + 1$
- $P(G)$ is even number
  - Make new bus route. Which is $v_i v_n$.

    $\lceil P(G+v)/2 \rceil = \lceil P(G)/2 \rceil + 1$.

Bus Route

- $P(G+v) = P(G) + 1$
- $P(G)$ is even number
  - Make new bus route. Which is $v_i v_n$.
    $\lceil P(G+v)/2 \rceil = \lceil P(G)/2 \rceil + 1$.
- $P(G)$ is odd number
  - Let's express each bus route as (v_l, v_m), v_l, and v_m are end stop of bus route. $P(G)$ is odd number, so least one pair of bus route is not pair of degree 1 stop. WLOG, at (v_l, v_m), v_m is not degree 1 vertex. We can find new bus route of (v_m, v_n), combine them. If route is duplicate, cut duplicate section. we are done.

Bus Route

- Given graph is tree. Because, there is no cycle route.

Bus Route

- Given graph is tree. Because, there is no cycle route.
- By our theorem, the answer is $\lceil P(G)/2 \rceil$

Bus Route

- Given graph is tree. Because, there is no cycle route.
- By our theorem, the answer is $\lceil P(G)/2 \rceil$
- Count each degree of stop, find $P(G)$, and calculate $\lceil P(G)/2 \rceil$.

Bus Route

- Given graph is tree. Because, there is no cycle route.
- By our theorem, the answer is $\lceil P(G)/2 \rceil$
- Count each degree of stop, find $P(G)$, and calculate $\lceil P(G)/2 \rceil$.
- Time complexity is $O(N)$. Space complexity is $O(N)$.

Clock

- # Submissions: 44(Onsite)
- # Accepted: 11(Onsite)
- First Solver: 한승헌(Onsite), clrmt(Open)
- Proposed by: Seoyoon Lee

Clock

second hand

- it goes 360 degrees every 60 seconds.

So the second hand rotate $6 * s$ degree in clockwise from the (0,1)

Clock

second hand

- it goes 360 degrees every 60 seconds.
- Second hand Rotate $360/60 = 6$ degrees per second.

So the second hand rotate $6 * s$ degree in clockwise from the (0,1)

Clock

minute hand

- it goes 360 degrees every 60 minutes.

So the minute hand at (0,1) rotate $6 * m + s/10$ degree in clockwise from (0,1)

Clock

minute hand

- it goes 360 degrees every 60 minutes.
- Minute hand Rotate $360/60 = 6$ degrees per minute,

So the minute hand at (0,1) rotate $6 * m + s/10$ degree in clockwise from (0,1)

Clock

minute hand

- it goes 360 degrees every 60 minutes.
- Minute hand Rotate $360/60 = 6$ degrees per minute,
- Minute hand Rotate $(6/60) = 1/10$ degree per second.

So the minute hand at (0,1) rotate $6 * m + s/10$ degree in clockwise from (0,1)

Clock

hour hand

- $360/12 = 30$ degrees per hour

Clock

hour hand

- $360/12 = 30$ degrees per hour
- $(30/60) = 0.5$ degrees per minute

Clock

hour hand

- $360/12 = 30$ degrees per hour
- $(30/60) = 0.5$ degrees per minute
- $1/120$ degrees per second

Clock

hour hand

- $360/12 = 30$ degrees per hour
- $(30/60) = 0.5$ degrees per minute
- $1/120$ degrees per second
- $(h * 30 + m * 0.5 + s/120)$ degree in clockwise from the (0,1)

Clock

- calculate minimum degree that any two line of three line(OA,OB,OC) forms, so calculate difference of degree that hour hand or minute hand or second hand ,

Clock

- calculate minimum degree that any two line of three line(OA,OB,OC) forms, so calculate difference of degree that hour hand or minute hand or second hand ,
- if difference is bigger than 180, subtract 360 by difference.

Clock

careful

- Set input variable as double

Clock

careful

- Set input variable as double
- `print("%.6f")`

Clock

careful

- Set input variable as double
- `print("%.6f")`
- `cout << setprecision(6) << fixed;`

Ctrl_CV

- # Submissions: 4(Onsite)
- # Accepted: 1(Onsite)
- First Solver: 한동규(Onsite), xiaowuc1(Open)
- Proposed by: Seoyoung Park

Ctrl_CV

There are two representative methods to solve this problem. To handle the string with maximum size of 200,000 we need to use the algorithm with time complesity at most $O(N \log^2 N)$.

- SA & LCP
- Binary Search & Hashing

Ctrl_CV

To avoid terrible time complexity $O(N^3)$, we can use LCP to find the longest common partial array.

- Suffix Array: $O(N \log N)$
- LCP Array: $O(N)$
- total: $O(N \log N)$

Ctrl_CV

- Suffix Array is sorted array include all suffixes of given word. $O(N \log N)$ algorithm for construct SA is well-known.
- Combined with LCP (Longest Common Prefix), we can find the longest common partial string. With SA, we can construct LCP in $O(N)$

Ctrl_CV

Another nice method is using Hashing. Reinterpret the problem as find the common string with length L.

- If there is common string with length L, There should be common string with length (L-1). Therefore we can use the method of binary search. $O(\log N)$
- Then, the problem can be interpreted to find the common string with length L. We can solve this kind of problem efficiently use Hash.

Ctrl_CV

However, there is another condition we must concerned: the longest common string appear disjoint in given string.

Thus, It is necessary to declare if the min index and max index stay away in original string.

Studytime

- # Submissions: 5(Onsite)
- # Accepted: 1(Onsite)
- First Solver: 한동규(Onsite), tpdnjs94(Open)
- Proposed by: Jongseo Lee

Studytime

This problem is typical type of 0,1 knapsack problem.

- i: number of subject

Studytime

This problem is typical type of 0,1 knapsack problem.

- i: number of subject
- r: required study time

Studytime

This problem is typical type of 0,1 knapsack problem.

- i: number of subject
- r: required study time
- Impo: function that output possible maximum importance

Studytime

This problem is typical type of 0,1 knapsack problem.

- i: number of subject
- r: required study time
- Impo: function that output possible maximum importance
- Impo(i, r) for situation that have to choose ith subject or not,if permitted maximum study time is r, output possible maximum importance

Studytime

- Situation that not select ith subject

Studytime

- Situation that not select ith subject
- $temp1 = Impo(i+1, r)$ the situation that before containing ith subject, put ith subject

Studytime

- Situation that select ith subject

Studytime

- Situation that select ith subject
- condition: $r >= requiredtime[i]$

Studytime

- Situation that select ith subject
- condition: $r >= requiredtime[i]$
- $temp2 = Impo(i-1, r - study)time[i]) + value[i]$ the situation that before containing ith subject, put ith subject

Studytime

- Situation that select ith subject
- condition: $r >= requiredtime[i]$
- $temp2 = Impo(i-1, r - study)time[i]) + value[i]$ the situation that before containing ith subject, put ith subject
- Select maximu value of temp1 and temp2