VE477 HW 5

Wu Jiayao 517370910257

October 31, 2019

1 The partition problem

1.1 Explanation

Omit

1.2 A simple strategy

It is not a good solution. A counter example is partition 1,2, 3,4 for 1,2,3,4. It doesn't solve the partition problem.

1.3 Recursive Algorithm

Denote s_i the size of i^{th} group in the partition. Transform the problem into find the minium value of the larger one between the sum of the k^{th} group during the partition and $M(n-s_k, k-1)$ for the $k-1$ group before. Then recursively, $M(n-s_k, k-1)$ is to calculated between the $k-1^{th}$ group and $k-2$ groups before.

1.4 Complexity

There is a total of $k \times n$ combinations. Trasversing takes at most n^2. Therefore,

$$T(n) = \mathcal{O}(N^3)$$

1.5 Stored quantites

Denote $s[i]$ the element in the set. $p[i] = \sum_{k=1}^{i} s_k$ could be stored.

1.6 Dynamic Programming

Input : A given arrangement S of non-negative numbers $s_1, s_2, \cdots, s_n$ and an integer k.
Output: Partition S into k ranges, so as to minimize the maximum sum over all the ranges

```
p = []
p[0] = 0
for i = 1 to n do
    p[i] = s_i + p[i − 1]
for i = 1 to n do
    M[i, 1] = p[i]
for j = 1 to k do
    M[1, j] = s_1
for i = 2 to n − 1 do
    for j = 2 to k do
        M[i, j] = ∞
        for x = 1 to i − 1 do
            s = max(M[x, j − 1], p[i] − p[x])
            if M[i, j] > s then
                M[i, j] = s
                D[i, j] = x
return M[n, k]
```

1.7 Proof of correctness

For $n = 1$ or $k = 1$, all the value in matrix M is correct. If $M[x, j-1]$ is correct, $M[x, j]$ is sure to be correct, and will not affect the value of $M[x, j-1]$. Hence, the algorithm is correct.

1.8 Time complexity

As written above, there are at most $k \times n \times n$ times of operations. The time complexity is $\mathcal{O}(kn^2)$.

1.9 Where to set the dividers

As already implemented above, the last position of partition is $D[n, k]$. Then combine with $M[n, k]$ to search from the back to find the value of the $k - 1^{th}, k - 2^{th}, \cdots, 1^{st}$ partition.

2 Critical Thinking

We can get a generator which generates the range $0 - 24$ by $5 \times B() + B()$. Similarly, the range $0 - 624$ can be generated. Therefore, the range $0 - 5^{2^n}, n \in \mathcal{Z}^+$ can be generated.

Input : B()
Output: A random number in the range $0 - n$

i satisfies $5^{2^{i-1}} < n+1 <= 5^{2^i}$
Let $B_i()$ be the related generator.
$num = \lfloor 5^{2^i}/(n+1) \rfloor \cdot (n+1)$
while $G = B_i() > num$ **do**
    continue
return G % $(n+1)$

3 Bellman-Ford algorithm

Input : A directed weighted graph $G = \langle V, E \rangle$
Output: Whether there exists negative cycles

Randomly choose a vertex s.
foreach $v \in V$ **do**
    if v *is* s **then**
        $dist[v] = 0$
    else
        $dist[v] = \infty$
for $i = 2$ *to* $|V|$ **do**
    foreach $(u, v) \in E$ **do**
        $tmp = dist[u] + weight(u, v)$
        if $tmp < dist[v]$ **then**
            $dist[v] = tmp$
for $i = 2$ *to* $|V|$ **do**
    foreach $(u, v) \in E$ **do**
        $tmp = dist[u] + weight(u, v)$
        if $tmp < dist[v]$ **then**
            return True
return False

4 Augmenting path

Omitted.

5 Wifi Problem

Algorithm 1: $WIFI(r, l, loc, pos)$

Input : $r,\ l$ *, locations of k hostpots, positions of n users*

Output: Whether all the users can connect to Wifi

Create an empty graph $G\langle V, E\rangle$

$V \leftarrow V\ +\ \{s, t\}$

for i *in range*$(0, n)$ **do**

  for j *in range*$(0, k)$ **do**

    if *Distance(user i, hostpot j)* $< r$ **then**

      Add the edge $(u[i], h[j])$to E with capacity of 1

for i *in range*$(0, n)$ **do**

  Add the edge $(s, u[i])$ to E with capacity of 1

for i *in range*$(0, k)$ **do**

  Add the edge $(h[i], t)$ to E with capacity of l

$F \leftarrow Edmonds - Karp(G)$

return $F\ ==\ n$

The time complexity is the same as Edmonds-Karp algorithm, which is $\mathcal{O}(|V||E|^2)$. Capacity of $(s, u[i])$ and $(u[i], h[j])$ is 1 means that one user can only access to one hostpot. Capacity of $(u[i], h[j])$ is l restircts the max number of users . The graph is constructed that the condition of max flow happens when all the users are connected to Wifi. Hence, it is correct.