Solutions for Week 1

Bao Jinge

1

As two kinds of buses stop at the bus stop every half hours, so we can just image a scene with an hour span. In this scene(which time interval is $[0, 60)$), every kind of buses can stop only twice. Here, we suppose without loss of generality as following
1. the first kind of buses(go to southwards) will stop at timestep a_1 and a_2 and the second buses(go to northwards) will stop at timestamp b_1 and b_2. $a_1, a_2, b_1, b_2 \in [0, 60)$.
2. $a_1 < b_1$, so $a_2 < b_2$ too.
3. Every month has only 30 days.
Here are equations

$$\frac{(a_1 - 0) + (a_2 - b_1) + (60 - b_2)}{(b_1 - a_1) + (b_2 - a_2)} = \frac{27}{3}$$

$$b_1 - a_1 = b_2 - a_2$$

$$(60 - b_2) + (a_1 - 0) = a_2 - b_1$$

$$a_2 - a_1 = b_2 - b_1 = 30$$

We got answer

$$b_1 - a_1 = b_2 - a_2 = 27$$

which means that the second kinds of bus(go south) always arrives the stop 27 minutes after the first kinds of bus(go north). So the person only only go south with probility of 27/30 and go north with probility of 3/30. Thus he visits his parents only three times per month.

2

(a)
1.When $n = 2^k$,

$$randbit(n) = \sum_{i=0}^{k-1} 2^i randbit()$$

2.When n is not a power of 2, we can write n as $n = \sum_{i=0} = a_i * 2^i$, which $a_i \in 0, 1$ So we get

$$randbit(n) = \sum_{i=0} a_i \cdot randbit(2^i)$$

(b) In C++, we can get $randbit()$ as follows

```
    // C++ code
    int randbit() {
        srand(time(NULL))
        int random = rand() % 2; //rand() is included in cstdlib
        return random;
    }
```

After running code, we got the length of the longest streak in my array a is 7.

3

After running codes, we got the minimum cut size is 3, and i need nearly 25 iterations of the subroutine to detect this. In codes, i use Disjoint Set Union algorithm. Some codes are as follows

```
    while (true) {
    int edge[MAXV][MAXV];
    vector<int> v;
    v.clear();
    for (int i = 0; i < nV; i++) {
        for (int j = 0; j < nV; j++) {
            edge[i][j] = _edge[i][j];
        }
    }
    for (int i = 0; i < nV; i++) {
        fa[i] = i;
    }
    for (int i = 0; i < nV; i++) {
        v.push_back(i);
    }
    while(v.size() > 2) {
        int index_i = rand() % v.size();
        int index_j = rand() % v.size();
        int fa_i = getfa(v[index_i]);
        int fa_j = getfa(v[index_j]);
        if(fa_i != fa_j) {
            if (edge[fa_i][fa_j] >= 1) {
                if (fa_i != fa_j) {
                    fa[fa_j] = fa_i;
                    for (int k = 0; k < nV; k++) {
                        fa[k] = getfa(fa[k]);
                    }
                    for (int k = 0; k < nV; k++) {
                        edge[fa_i][k] = edge[k][fa_i] = edge[fa_i][k] + edge[fa_j][k];
                    }
                    edge[fa_i][fa_i] = 0;
                    v.erase(v.begin() + index_j);
                }
            }
        }
    }
//  cout << "the minimum cut size:" << edge[fa[v[0]]][fa[v[1]]] << endl;
//  cout << "iterations of the subroutine need to detect this answer" << ++iter << endl;
    min_cut_size = min(min_cut_size, edge[fa[v[0]]][fa[v[1]]]);
```