طراحی الگوریتم ها

نیم‌سال دوم ۹۸ـ۹۹

دانشکده‌ی مهندسی کامپیوتر گردآورندگان: اسرا کاشانی ، کیارش گل زاده ، محمدکاظم فقیه ، آراد محمدی

پاسخ نامه تمرین عملی سری اول — مقدمات cpp و الگوریتم های حریصانه

مسئله‌ی ۱. سروش لشکری

به ازای هر صف نانوایی، یک queue در نظر می‌گیریم. (یعنی آرایه‌ای از queue ها داریم.) هر فردی که به انتهای صف k ام اضافه می‌شود، شعور آن فرد را در انتهای صف k ام push می‌کنیم. همچنین، یک set از pair<int , int> به نمایندگی افرادی که سرِ صف‌ها قرار دارند، در نظر می‌گیریم(set در زبان cpp به صورت red-black tree است). به ازای افراد سر صف، یک pair که عدد اول آن، شعور آن فرد، و عدد دوم آن، شماره‌ی صفی است که فرد در آن قرار دارد، در set نگه می‌داریم. حال، به ازای کوئری‌های «+»، علاوه بر push کردن در صف، اگر صف مورد نظر خالی باشد، یعنی فردِ جدید به محض وارد شدن در سر صف قرار می‌گیرد! پس pair متناظر با او را در set اضافه می‌کنیم. به ازای کوئری‌های «؟»، عضو ماکسیمم set (عضو با بیشترین شعور، از میان افرادی که سر صف هستند) را پیدا می‌کنیم و شعور او را چاپ می‌کنیم. (از آن‌جایی که شعور افراد متمایز هستند، این عضو یکتا است.) لازم به ذکر است که در مقایسه‌ی pair ها، ابتدا مؤلفه‌ی اول مورد مقایسه قرار می‌گیرد، و در صورت تساوی، مؤلفه‌ی دوم مورد مقایسه قرار می‌گیرد. حال، این فرد را از صفی که در آن قرار دارد pop کرده، و از set آن را حذف می‌کنیم. حال اگر فردی پشت سر این فرد که تازه خارج شده‌است باشد، اکنون سر صف است، و باید آن را به set اضافه کنیم. اگر به ازای یک کوئری «؟»، set خالی باشد، باید عبارت empty all را چاپ کنیم.
کد جواب سوال را می توان در زیر مشاهده کرد.

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  constexpr int MAX_K = 1e5;
5  queue<int> queues[MAX_K + 1];
6  set<pair<int, int>, greater<pair<int, int> > > front_of_queues;
7
8  int main()
9  {
10     int q, k;
11     cin >> q >> k;
12
13     while (q--)
14     {
15         char c;
```

```
16          cin >> c;
17          if (c == '+')
18          {
19              int p, s;
20              cin >> p >> s;
21              s--;
22
23              if (queues[s].empty())
24                  front_of_queues.insert({p, s});
25              queues[s].push(p);
26          }
27          else
28          {
29              if (front_of_queues.empty())
30              {
31                  cout << "all empty" << endl;
32                  continue;
33              }
34              pair<int, int> mx = *front_of_queues.begin();
35              cout << mx.first << endl;
36              front_of_queues.erase(front_of_queues.begin());
37              queues[mx.second].pop();
38              if (!queues[mx.second].empty())
39                  front_of_queues.insert({queues[mx.second].front(),
40                  mx.second});
41          }
42      }
43
44      return 0;
45  }
```

مسئله‌ی ۲. امتحانات

برای حل این سوال کافیست که از الگوریتم حریصانه استفاده کنیم.بعد از اینکه آرایه D[i] و T[i] ها را بر حسب ورودی مسئله به دست آوردیم، یک صف اولویت دارد درست می کنیم و آن ها را بر این صف push می کنیم.بعد این صف را بر حسب عنصر اول آن ها مرتب می کنیم و سپس روی این صف مرتب شده iterate می کنیم.برای هر کدام از امتحان ها نگاه میکنیم که روز های خالی تا روز امتحان چقدر است.اگر روز های خالی که تا الان داریم ، از مقدار مورد نیاز خواندن آن امتحان بیشتر بود یکی به مقدار امتحان هایی که می تواند پاس کند اضافه می کنیم و اگر نبود ، وارد if اول می شویم که اگر آخرین امتحانی که پاس کردیم ، زمان بیشتر از این امتحان نیاز داشت ، آن امتحان را از امتحان های پاس شده کم می کنیم و روند را ادامه می دهیم.کد این قسمت در زیر آمده است.

```
1  vector<pair<long long, long long>> exams;
```

```
2
3       for (int i = 0; i < N; i++)
4           exams.emplace_back(D[i], T[i]);
5
6       sort(exams.begin(), exams.end());
7       priority_queue<long long> pq;
8       int pass_count = 0;
9       long long current_day = 0, free_days = 0;
10
11      for (pair<long long, long long> &exam : exams) {
12          long long D = exam.first;
13          long long T = exam.second;
14
15          free_days += D - current_day;
16          current_day = D;
17
18          if (free_days < T && !pq.empty() && pq.top() > T) {
19              free_days += pq.top();
20              pass_count--;
21              pq.pop();
22          }
23
24          if (free_days >= T) {
25              free_days -= T;
26              pass_count++;
27              pq.push(T);
28          }
29      }
30      cout << pass_count;
```

موفق باشید :)