Aufgabe 2: Vollgeladen

Team-ID: 00067

Team-Name: Panic! at the Kernel

Bearbeiter/-innen dieser Aufgabe:
Christopher Besch

21. November 2021

Inhaltsverzeichnis

1 Abstract

Es werden zwei Tabellen erstellt:

- Eine mit der minimalen Bewertung des optimalen Weges zu jedem Hotel nach d Tagen und
- eine mit dem jeweiligen Vorgänger auf diesem optimalen Weg.

Anschliessend werden diese beiden Tabellen benutzt, um den optimalen Weg zum Ziel zu bestimmen. Wenn dieser existiert, wird die minimale Bewertung und die besuchten Hotels berechnet. Schlussendlich wird die Lösungsidee in C++ implementiert und Spezialfälle mit Beispieleingaben erläutert.

2 Lösungsidee

Für jedes Hotel x lässt sich sagen, ob es innerhalb von d Tagen erreicht werden kann. Ist dies für ein d der Fall, kann die minimale Bewertung aller Hotels, die auf dem optimalen Weg zu x liegenden, angegeben werden. Dieser optimale Weg benötigt genau d Tage. Er ist in dem Sinne optimal, dass kein anderer Weg existiert, der in d Tagen das Hotel erreicht und eine besserer minimale Bewertung aufweist. Daher weist der optimale Weg immer die **beste minimale Bewertung** aller Hotels auf diesem Weg auf.

Diese Werte können in einer Tabelle—der Minimumstabelle—dargestellt werden und werden minimale Webbewertung genannt:

```
                  min rating at day:
idx     rating    0     1       2       3       4       5
===     ======    =     =       =       =       =       =
0       inf       inf
1       4.3             4.3
2       4.8             4.8     4.3
3       2.7             2.7     2.7     2.7
4       2.6             2.6     2.6     2.6     2.6
5       3.6                     3.6     3.6     2.7     2.6
```

```
   6       0.8                     0.8     0.8     0.8     0.8
11 7       4.4                     2.7     3.6     3.6     2.7
   8       2.8                             2.7     2.8     2.8
13 9       2.6                             2.6     2.6     2.6
   10      2.1                                     2.1     2.1
15 11      2.8                                     2.7     2.8
   12      3.3                                             2.7
17 13      inf                                             2.7
```

Neben den Hotels werde hier mit dem Index 0 der Startort und mit 13 das Ziel gelistet. Dessen Bewertung ist nie schlechter als die eines beliebigen Hotels, weshalb sie als unendlich angesehen wird. Da eine Fahrt, die länger als fünf Tage dauert nicht zulässig ist, werden diese Optionen weggelassen.

Anhand dieser Daten lässt sich recht leicht die minimale Bewertung auf dem Weg zu jedem beliebigen Hotel ausgeben. Interessant ist dieser Wert für das Ziel. Es muss lediglich das Minimum aller Werte in der letzten Zeile bestimmt werden. Die Reise endet dann am sogenannten optimalen Tag.

Um nun die einzelnen Hotels, die optimal zum Ziel führen, zu bestimmen, wird eine zweite Tabelle—die Vorgängertabelle—benötigt. Diese weist die gleichen Dimensionen auf, gibt allerdings für jedes Hotel x und Tag d das Hotel, das am vorherigen Tag $(d-1)$ zuletzt besucht wurde, an. So kann vom Ziel am optimalen Tag ausgehend immer das zuletzt besuchte Hotel bestimmt werden. Dieser Vorgang endet, wenn man am Startort angekommen ist.

2.1 Berechnung der Tabellen

Als Taglänge wird die Strecke verstanden, die an einem Tag zurückgelegt werden kann. Zu jedem Hotel x kann man am Tag d von allen Hotels, die maximal eine Taglänge vor x liegen und innerhalb von genau $d-1$ Tagen erreichbar sind, gelangen. Diese Hotels bilden die Menge Y. Alle Hotels, die hinter x liegen, sind nicht zu verwenden.

Die minimale Bewertung a zum Hotel x nach d Tagen entspricht mit

- dem Minimum b der minimalen Wegbewertungen zu allen Hotels aus Y für den Tag $d-1$ und
- der Bewertung c des Hotels x:

$$a = min(b, c)$$

a wird in der Minimumstabelle gespeichert. Der entsprechende Wert in der Vorgängertabelle entspricht dem Index des Hotels dessen minimale Wegbewertung gewählt wurde.

Da so immer nur die Information von Hotels, die vor dem aktuellen liegen, benötigt werden, kann das Prinzip der dynamischen Programmierung verwendet werden. Hierbei werden die Tabellen anfangend beim Startort Hotel für Hotel gefüllt. Jedes weitere Hotel benötigt ausschliesslich die bereits berechnete Information.

3 Umsetzung und Quellcode

Die Lösungsidee wird in C++ implementiert. Zur Repräsentation der Tabellen werden mehrdimensionale vectors benutzt:

```
// min_ratings[x][y] -> min rating till hotel x requiring y days to get to
// -1 -> impossible
std::vector<std::vector<float>> min_ratings(n + 2, std::vector<float>(DAYS + 1, -1));
std::vector<std::vector<int>>   last_hotel(n + 2, std::vector<int>(DAYS + 1, -1));
```

`DAYS` entspricht der Anzahl an Tagen, die die Reise maximal dauern darf. Diese Konstante wird an allen relevanten Stellen respektiert, weshalb leicht eine Berechnung für mehr (oder weniger) als fünf Tage durchgeführt werden kann.

Nun kann mit der Funktion `populate_tables` diese Tabellen der Lösungsidee nach füllen.

```
void populate_tables(
    const std::vector<std::pair<int, float>>& hotels,
    std::vector<std::vector<float>>&          min_ratings,
    std::vector<std::vector<int>>&            last_hotel)
{
```

```
    // the "first" (virtual) hotel never has the worst rating
    min_ratings[0][0] = inf;
    // go through all but "first" hotels
    for(auto cur = hotels.begin() + 1; cur != hotels.end(); ++cur) {
        int cur_idx = cur - hotels.begin();
        // first hotel 'cur' can be reached from
        auto prev = std::lower_bound(hotels.begin(), hotels.end(),
                                     std::make_pair(cur->first - DAY_LEN, .0f));
        for(; prev != cur; ++prev) {
            int prev_idx = prev - hotels.begin();
            // go through past days when prev can be reached
            // call ride off after 'DAYS' days
            for(int prev_day = 0, cur_day = 1; prev_day < DAYS; ++prev_day, ++cur_day) {
                // if the hotel 'prev' can't be reached in 'prev_day' days
                if(min_ratings[prev_idx][prev_day] == -1)
                    continue;
                float new_min = std::min(min_ratings[prev_idx][prev_day], cur->second);
                // update when better
                if(new_min > min_ratings[cur_idx][cur_day]) {
                    min_ratings[cur_idx][cur_day] = new_min;
                    last_hotel[cur_idx][cur_day]  = prev_idx;
                }
            }
        }
    }
}
```

Diese Funktion ist die Aufwändigste der Implementation. Obwohl sie drei verschachtelte Schleifen enthält ist die Laufzeit der gesamten Implementation für fünf Tage linear, wenn davon ausgegangen wird, dass der Abstand zwischen zwei Hotels in etwa immer gleich ist. Durch diese Annahme benötigen die inneren Schleifen unabhängig von der Anzahl an Hotels immer gleich lange. Es bleibt nur die Äussere.

Wenn die Tabellen produziert wurde, können mit `get_best_day` die Werte des „Zielhotels“ in der Minimumstabelle linear nach dem Optimum durchsucht werden. Wenn es nicht möglich ist, unter den gegebenen Anforderungen das Ziel zu erreichen, wird -1 zurückgegeben.

```
int get_best_day(const std::vector<std::vector<float>>& min_ratings, int n)
{
    int   best_day {-1};
    float best_min_rating {-1};
    for(int i = 0; i < DAYS + 1; ++i) {
        if(min_ratings[n + 1][i] > best_min_rating) {
            best_min_rating = min_ratings[n + 1][i];
            best_day        = i;
        }
    }
    return best_day;
}
```

Wurde festgestellt, dass es einen Weg zum „Zielhotel“ innerhalb von fünf Tagen gibt, kann der optimale Weg mithilfe der Funktion `construct_path` bestimmt werden:

```
void construct_path(
    const std::vector<std::vector<int>>& last_hotel,
    int                                  best_day,
    int                                  n,
    std::vector<int>&                    path)
{
    int cur_idx = last_hotel[n + 1][best_day];
    for(int cur_day = best_day - 1; cur_day; --cur_day) {
        path.push_back(cur_idx);
        cur_idx = last_hotel[cur_idx][cur_day];
    }
    std::reverse(path.begin(), path.end());
}
```

Hierbei wird `cur_idx` immer so aktualisiert, dass immer das jeweilig vorhergehende Hotel referenziert wird. Die Suche wird beendet, wenn der Startort erreicht wurde.

Zum Schluss werden die beiden Tabellen und der finale, optimale Weg ausgegeben.

4 Beispiele

Das Pythonscript `tabs.py` wird benutzt, um tabs korrekt zu formatieren. Da es für die Lösung der Aufgabe nicht von Nöten ist, ist es in der Abgabe nicht enthalten. Es kann hier heruntergeladen werden.

Neben dem optimalen Weg wird ebenfalls die Minimumstabelle abgedruckt. Die + Zeichen zeigen, dass dieses Hotel im optimalen Weg enthalten ist.

Hotels 1

```
./tabs.py <(cat examples/hotels1.txt | ./a.out)

                          min rating at day:
idx     distance rating   0     1       2       3       4       5
===     ======== ======   =     =       =       =       =       =
0       0        inf      inf
1       12       4.3            4.3
2       326      4.8            4.8     4.3
3       347      2.7    +       2.7     2.7     2.7
4       359      2.6            2.6     2.6     2.6     2.6
5       553      3.6                    3.6     3.6     2.7     2.6
6       590      0.8                    0.8     0.8     0.8     0.8
7       687      4.4    +               2.7     3.6     3.6     2.7
8       1007     2.8    +                       2.7     2.8     2.8
9       1008     2.6                            2.6     2.6     2.6
10      1321     2.1                                    2.1     2.1
11      1360     2.8    +                               2.7     2.8
12      1411     3.3                                            2.7
13      1680     inf                                            2.7

These hotels should be used with min rating 2.7:
idx     location        rating  min rating till here
===     ========        ======  ====================
3       347             2.7     2.7
7       687             4.4     2.7
8       1007            2.8     2.7
11      1360            2.8     2.7
```

Hotels 2

```
./tabs.py <(cat examples/hotels2.txt | ./a.out)

                          min rating at day:
idx     distance rating   0     1       2       3       4       5
===     ======== ======   =     =       =       =       =       =
0       0        inf      inf
1       340      1.6            1.6
2       341      2.2            2.2     1.6
3       341      2.3    +       2.3     2.2     1.6
4       342      2.1            2.1     2.1     2.1     1.6
5       360      1.9            1.9     1.9     1.9     1.9     1.6
6       361      4.4                    2.3     2.2     2.1     1.9
7       362      3.1                    2.3     2.3     2.2     2.1
8       442      5                      2.3     2.3     2.3     2.2
9       567      4.9                    2.3     2.3     2.3     2.3
10      700      3      +               2.3     2.3     2.3     2.3
11      710      2.9                    1.9     2.3     2.3     2.3
12      718      1.4                    1.4     1.4     1.4     1.4
13      987      4.6                            2.3     2.3     2.3
14      1051     2.3    +               2.3     2.3     2.3     2.3
15      1053     4.8                            2.3     2.3     2.3
16      1057     0.2                            0.2     0.2     0.2
17      1199     5                              2.3     2.3     2.3
18      1279     5                              2.3     2.3     2.3
19      1367     4.5                            2.3     2.3     2.3
20      1377     1.8                            1.8     1.8     1.8
21      1377     1.6                            1.6     1.6     1.6
22      1377     2                              2       2       2
23      1378     2.1                            2.1     2.1     2.1
```

```
24      1378     2.2                                      2.2     2.2
25      1380     5       +                                2.3     2.3
26      1737     inf                                              2.3

These hotels should be used with min rating 2.3:
idx     location        rating  min rating till here
===     ========        ======  ====================
3       341             2.3     2.3
10      700             3       2.3
14      1051            2.3     2.3
25      1380            5       2.3
```

Hotels 3

```
./tabs.py <(cat examples/hotels3.txt | ./a.out)

These hotels should be used with min rating 0.3:
idx     location        rating  min rating till here
===     ========        ======  ====================
97      358             2.5     2.5
196     717             0.3     0.3
297     1075            0.8     0.3
401     1433            1.7     0.3
```

Aufgrund von 500 Hotels wird für dieses Beispiel die Minimumstabelle nicht abgedruckt.

Hotels 5

```
./tabs.py <(cat examples/hotels5.txt | ./a.out)

These hotels should be used with min rating 5:
idx     location        rating  min rating till here
===     ========        ======  ====================
242     280             5       5
581     636             5       5
913     987             5       5
1168    1271            5       5
```

Hier wird die Minimumstabelle ebenfalls weggelassen.

Unmögliches Beispiel

```
./tabs.py <(cat examples/my_hotels0.txt | ./a.out)

                          min rating at day:
idx     distance rating   0     1       2       3       4       5
===     ======== ======   =     =       =       =       =       =
0       0        inf      inf
1       1680     inf

It is impossible to reach the destination in 5 days without draining the phones' batteries.
Maybe read a book instead.
```

Dieses Beispiel enthält kein einziges Hotel, weshalb die gesamte Fahrzeit von 1680 Minuten nicht ohne geladene Smartphones überstanden werden kann.

Mögliches Beispiel nach sechs Tagen

```
./tabs.py <(cat examples/my_hotels1.txt | ./a.out)

                          min rating at day:
idx     distance rating   0     1       2       3       4       5
===     ======== ======   =     =       =       =       =       =
0       0        inf      inf
1       300      5              5
2       600      5                      5
3       900      5                              5
4       1200     5                                      5
5       1500     5                                              5
6       1800     inf

It is impossible to reach the destination in 5 days without draining the phones' batteries.
Maybe read a book instead.
```

Es zeigt sich, dass die Hotels geradeso nicht ausreichen, um das Ziel innerhalb von fünf Tagen zu erreichen. Wenn allerdings die Konstante DAYS auf 6 gesetzt wird, ist das Beispiel lösbar:

```
./tabs.py <(cat examples/my_hotels1.txt | ./a.out)

                          min rating at day:
idx     distance rating   0     1       2       3       4       5       6
===     ======== ======   =     =       =       =       =       =       =
0       0        inf      inf
1       300      5      +       5
2       600      5      +               5
3       900      5      +                       5
4       1200     5      +                               5
5       1500     5      +                                       5
6       1800     inf                                                    5

These hotels should be used with min rating 5:
idx     location        rating  min rating till here
===     ========        ======  ====================
1       300             5       5
2       600             5       5
3       900             5       5
4       1200            5       5
5       1500            5       5
```