Dappy

BWINF Aufgabe 5

Team ID: 00871

Definitionen

Punkt := Tourpunkt
Segment := Aufeinanderfolgende Teilstrecke der Tour wobei genau nur der Anfang und das Ende essenziell sind. Ausgenommen von dieser Regel sind der Anfangs und Endpunkt der kompletten Tour - irrelevant davon ob sie essenziell sind bilden sie, respektiv, den Anfang oder Ende eines Segments.
Loop := Aufeinanderfolgende Teilstrecke der Tour wobei der Anfang und das Ende dieser Teilstrecke den gleichen Ort haben und außerdem der komplette Loop in einem einzigen Segment sein soll.
simpler Loop := Ein Loop der außerdem keinen Punkt besitzt, der den gleichen Ort wie der Anfang und das Ende besitzt

Bemerkung

Es sollte kurz angemerkt werden, dass ich streng nach der Aufgabenstellung gegangen bin - der Anfangs und Endpunkt wird hierbei nicht angepasst, da dies nicht eine Kürzung mittels Elimination einer geschlossenen Teiltour wäre. Außerdem sollte auffallen, dass eine solche extra Aufgabenstellung trivial ist - wenn wir einen beliebigen Startpunkt wählen können, solange wir keine essenziellen Punkt vernachlässigen, wählen wir einfach den ersten essenziellen Punkt, analog dazu wäre die Wahl des Endpunktes. Außerdem sollte angemerkt werden, das bei der Elimination einer geschlossenen Teiltour, die Anfangs und Endpunkte dieser Teiltour beide bestehen bleiben.

Idee

Bemerke zuerst: Lösen wir das Problem für ein beliebiges Segment optimal, lösen wir auch das gesamte Problem ideal. Betrachte also ein einzelnes Segment und versuche es ideal zu lösen:

Es sollte zuerst auffallen, dass wir das Problem nicht greedy lösen können: Löschen wir z.B. immer den ersten Loop den wir finden, könnte es bei Segmenten wie ABACDFGEB zu Problemen kommen. Da wir den Loop ABA löschen, haben wir nicht mehr die Chance den Loop BACDFGEB zu löschen, der länger sein könnte. Jedoch ist dieses Problem trivial mittels DP lösbar. Dafür muss man nur bemerken, dass dieses Problem im Prinzip das gleiche ist wie https://cses.fi/problemset/task/1140.

Die Lösungsidee ist wie folgt: Geh zuerst einmal durch das Segment und merke dir alle simplen Loops. Alle anderen Loops kann man nämlich immer durch Aneinanderreihungen von simplen Loops erreichen. Lass im folgenden Loop_i den i'ten Loop denotieren, welcher drei Attribute hat: Anfang denotiert den Index des Anfangs des Loops, Ende den Index des Ende des Loops und $\mathrm{Distanz}$ wie viel Distanz zwischen dem Anfang und dem Ende des Loops liegt.

Danach initialisiere ein dp Array mit der Länge des Segments. Folgende rekursive Formel gilt:

$$dp[x] = \begin{cases} max(dp[x-1], dp[\mathrm{Loop}_{i,\mathrm{Anfang}}] + \mathrm{Loop}_{i,\mathrm{Distanz}} & \text{falls } \mathrm{Loop}_{i,\mathrm{Ende}} = x \\ dp[x-1] & \text{sonst} \end{cases}$$

Mit dieser Idee kann man nicht nur das CSES Problem lösen, sondern auch das hier. Im Kontrast zum CSES Problem, müssen wir jedoch noch backtracken. Dazu starten wir am letzten Punkt des Segments und verfolgen mithilfe dieser zweiten rekursiven Formel und merken uns alle Punkte die wir besichtigen, bis wir am Anfang des Segments sind:

$$\mathrm{backtrack}(x) = \begin{cases} x-1 & \text{falls } dp[x] = dp[x-1] \\ \mathrm{Loop}_{i,\mathrm{Anfang}} \text{ wobei } \mathrm{Loop}_{i,\mathrm{Ende}} = x & \text{sonst} \end{cases}$$

Wenn wir uns gemerkt haben welche Punkte wir also benutzt haben, speichern wir dies alles ab und gehen weiter zum nächsten Segment.

Sind wird fertig mit allen Segmenten, brauchen wir nur noch eine passende Ausgabe zu den erarbeiteten Ergebnissen. Wir haben als Ergebnis ein Array von genau den Punkten die benutzt werden. Das einzige was wir somit wirklich anpassen müssen, ist die totale Distanz - dabei gehen wir einfach durch alle gemerkten Punkte und falls die letzten beiden Punkte übereinstimmen vom Ort her, sollte die Distanz auch 0 sein - hier haben wir nämlich eine geschlossene Teiltour weggelassen. Diese angepasste Distanz und das Ergebnis der besichtigten Punkte geben wir nun nur noch aus.

Implementation

Betrachten wir erst den Prozess wo wir alle Loops bestimmen wollen: Der triviale Ansatz wäre für jeden Punkt einmal komplett durch das Array gehen und überprüfen ob ein weiterer Punkt existiert mit dem gleichen Ort. Falls dies der Fall ist, speichern wir uns diesen Loop einfach ab. Das ist leider aber viel zu langsam, es ergibt sich eine Zeitkomplexität von $O(n^2)$. Jedoch können wir diesen Prozess auch in $O(n \cdot \log(n))$: Benutzen wir nämlich eine Map um zu überprüfen ob wir schon einen Punkt mit einem beliebigen Ort haben, ergibt sich $O(n \cdot \log(n))$. Den Anfangsindex speichern wir einfach unter der Map ab und da wir auch schon von der Eingabe ausgehend ein Präfix Array der Distanzen haben, können wir auch die Distanz zwischen Anfang und Ende in konstanter Zeit berechnen.

Nun der zweite Prozess: Die rekursive Formel braucht nämlich die Information ob ein Loop_i existiert welcher bei x endet. Dies könnten wir in linearer Zeit machen, mittels eines weiteren Arrays, wo wir diese Information unter $\mathrm{saved}[x]$ abspeichern, da wir aber eine asymptotisch Laufzeit von $O(n \cdot \log(n))$ jetzt schon haben, können wir auch einfach unsere Map von vorher weiterbenutzen und in $O(n \cdot \log(n))$ dies Information extrahieren.

Pseudocode zum bestimmen aller einfachen Loops:

```
seen := map die unter den Namen der besuchten Punkte den letzten Index abspeichert
foundEvents := array welches alle simplen Loops abspeichern

for point, index in segment:
        if seen.keys contains point.name:
                foundEvents append (index, seen[point.name], dist[index]-dist[seen[point.name]])

        seen[point.name] = index
```

Pseudocode zum dp:

```
for index in segment:
        bestesErgebnis = dp[index-1]
        falls bei index ein simpler Loop (anfangsindex, endeindex, distanz) endet:
                bestesErgebnis = maximum aus bestesErgebnis und dp[anfangsIndex] + distanz
        dp[index] = bestesErgenis
```

Laufzeit

Dies braucht insgesamt eine Zeitkomplexität von $O(k + n \cdot \log(n))$, wobei k die Anzahl von simplen Loops ist und n die Laenge des Segments. Da aber jeder Punkt des Segments maximal das Ende eines simplen Loops ist, gilt folgend $k \leq n$ und somit insgesamt eine Zeitkomplexität von $O(n \cdot \log(n))$

Beispieleingaben

Bemerkung

Die Ausgabe gibt ein bisschen mehr als nötig aus - für jedes Segment außerdem das DP Array. Dies kann teils interessant sein, deswegen lass ich es einfach drin - die eigentlich Ausgabe befindet sich darunter.

Das Ausgabeformat ist simpel: Zuerst wird genannt wie viel Distanz insgesamt gespart wurde und dann was der optimale Pfad ist. Zum beschreiben wird ein leicht anderes Format als die Eingabe benutzt:

```
{Punkt.name} during year {Punkt.year} with a total distance {prefix distance to Punkt} --- {bool indicating whether Punkt is essential}
```

tour1.txt

Eingabe

```
18
Brauerei,1613,X,0
Karzer,1665,X,80
Rathaus,1678,X,150
Gründungsstein,1685, ,310
Wallanlage,1690, ,410
Rathaus,1739,X,500
Euler-Brücke,1768, ,680
Fibonacci-Gaststätte,1820,X,710
Schiefes Haus,1823, ,830
Theater,1880, ,960
Emmy-Noether-Campus,1912,X,1090
Fibonacci-Gaststätte,1923, ,1180
Hilbert-Raum,1945, ,1300
Schiefes Haus,1950, ,1400
Gauß-Turm,1952, ,1450
Emmy-Noether-Campus,1998,X,1780
Euler-Brücke,1999, ,1910
Brauerei,2012, ,2060
```

Ausgabe

```
We have saved a total distance of 1040 using the following path:
Brauerei during year 1613 with a total distance of 0 --- True
Karzer during year 1665 with a total distance of 80 --- True
Rathaus during year 1678 with a total distance of 150 --- True
Rathaus during year 1739 with a total distance of 150 --- True
Euler-BrÃ¼cke during year 1768 with a total distance of 330 --- False
Fibonacci-GaststÃ¤tte during year 1820 with a total distance of 360 --- True
Schiefes Haus during year 1823 with a total distance of 480 --- False
Theater during year 1880 with a total distance of 610 --- False
Emmy-Noether-Campus during year 1912 with a total distance of 740 --- True
Emmy-Noether-Campus during year 1998 with a total distance of 740 --- True
Euler-BrÃ¼cke during year 1999 with a total distance of 870 --- False
Brauerei during year 2012 with a total distance of 1020 --- False
```

tour2.txt

Eingabe

```
18
Brauerei,1613, ,0
Karzer,1665,X,80
Rathaus,1678, ,150
Gründungsstein,1685, ,310
Wallanlage,1690, ,410
Rathaus,1739, ,500
```

```
Euler-Brücke,1768, ,680
Fibonacci-Gaststätte,1820,X,710
Schiefes Haus,1823, ,830
Theater,1880, ,960
Emmy-Noether-Campus,1912,X,1090
Fibonacci-Gaststätte,1923, ,1180
Hilbert-Raum,1945, ,1300
Schiefes Haus,1950, ,1400
Gauß-Turm,1952, ,1450
Emmy-Noether-Campus,1998,X,1780
Euler-Brücke,1999, ,1910
Brauerei,2012, ,2060
```

Ausgabe

```
We have saved a total distance of 1040 using the following path:
Brauerei during year 1613 with a total distance of 0 --- False
Karzer during year 1665 with a total distance of 80 --- True
Rathaus during year 1678 with a total distance of 150 --- False
Rathaus during year 1739 with a total distance of 150 --- False
Euler-BrÃ¼cke during year 1768 with a total distance of 330 --- False
Fibonacci-GaststÃ¤tte during year 1820 with a total distance of 360 --- True
Schiefes Haus during year 1823 with a total distance of 480 --- False
Theater during year 1880 with a total distance of 610 --- False
Emmy-Noether-Campus during year 1912 with a total distance of 740 --- True
Emmy-Noether-Campus during year 1998 with a total distance of 740 --- True
Euler-BrÃ¼cke during year 1999 with a total distance of 870 --- False
Brauerei during year 2012 with a total distance of 1020 --- False
```

tour3.txt

Eingabe

```
14
Talstation,1768, ,0
Wäldle,1805, ,520
Mittlere Alp,1823, ,1160
Observatorium,1833, ,1450
Wäldle,1841, ,1700
Bergstation,1866, ,2370
Observatorium,1874,X,2740
Piz Spitz,1898, ,3210
Panoramasteg,1912,X,3430
Bergstation,1928, ,3690
Ziegenbrücke,1935, ,3870
Panoramasteg,1952, ,4030
Ziegenbrücke,1979,X,4280
Talstation,2005, ,4560
```

Ausgabe

```
We have saved a total distance of 1890 using the following path:
Talstation during year 1768 with a total distance of 0 --- False
WÃ¤ldle during year 1805 with a total distance of 520 --- False
Mittlere Alp during year 1823 with a total distance of 1160 --- False
```

```
Observatorium during year 1833 with a total distance of 1450 --- False
Observatorium during year 1874 with a total distance of 1450 --- True
Piz Spitz during year 1898 with a total distance of 1920 --- False
Panoramasteg during year 1912 with a total distance of 2140 --- True
Panoramasteg during year 1952 with a total distance of 2140 --- False
ZiegenbrÃ¼cke during year 1979 with a total distance of 2390 --- True
Talstation during year 2005 with a total distance of 2670 --- False
```

tour4.txt

Eingabe

```
29
Blaues Pferd,1523, ,0
Alte Mühle,1544, ,110
Marktplatz,1549, ,210
Zeughaus,1559, ,310
Marktplatz,1562, ,410
Springbrunnen,1571, ,490
Dom,1596,X,560
Bogenschütze,1610, ,680
Marktplatz,1622, ,820
Ruhiges Eck,1625, ,850
Frauentor,1636, ,910
Ruhiges Eck,1651, ,980
Springbrunnen,1675, ,1000
Bogenschütze,1683, ,1070
Schnecke,1698,X,1220
Fischweiher,1710, ,1400
Reiterhof,1728,X,1520
Schnecke,1742, ,1660
Schmiede,1765, ,1830
Große Gabel,1794, ,1940
Schnecke,1829, ,2050
Europapark,1852, ,2280
Große Gabel,1874, ,2550
Fingerhut,1917,X,2620
Stadion,1934, ,2740
Marktplatz,1962, ,2830
Baumschule,1974, ,2910
Polizeipräsidium,1991, ,3050
Blaues Pferd,2004, ,3200
```

Ausgabe

```
We have saved a total distance of 1200 using the following path:
Blaues Pferd during year 1523 with a total distance of 0 --- False
Alte MÃ¼hle during year 1544 with a total distance of 110 --- False
Marktplatz during year 1549 with a total distance of 210 --- False
Marktplatz during year 1562 with a total distance of 210 --- False
Springbrunnen during year 1571 with a total distance of 290 --- False
Dom during year 1596 with a total distance of 360 --- True
BogenschÃ¼tze during year 1610 with a total distance of 480 --- False
BogenschÃ¼tze during year 1683 with a total distance of 480 --- False
Schnecke during year 1698 with a total distance of 630 --- True
```

```
Fischweiher during year 1710 with a total distance of 810 --- False
Reiterhof during year 1728 with a total distance of 930 --- True
Schnecke during year 1742 with a total distance of 1070 --- False
Schmiede during year 1765 with a total distance of 1240 --- False
GroÃŸe Gabel during year 1794 with a total distance of 1350 --- False
GroÃŸe Gabel during year 1874 with a total distance of 1350 --- False
Fingerhut during year 1917 with a total distance of 1420 --- True
Stadion during year 1934 with a total distance of 1540 --- False
Marktplatz during year 1962 with a total distance of 1630 --- False
Baumschule during year 1974 with a total distance of 1710 --- False
PolizeiprÃ¤sidium during year 1991 with a total distance of 1850 --- False
Blaues Pferd during year 2004 with a total distance of 2000 --- False
```

tour5.txt

Eingabe

```
42
Gabelhaus,1638, ,0
Burgruine,1654, ,140
Labyrinth,1667, ,220
Hängepartie,1672, ,470
Hexentanzplatz,1681, ,730
Gabelhaus,1699, ,820
Hexentanzplatz,1703,X,980
Eselsbrücke,1711, ,1100
Dreibannstein,1724, ,1210
Alte Wache,1733, ,1340
Palisadenhaus,1740, ,1490
Dreibannstein,1752, ,1620
Schmetterling,1760,X,1770
Dreibannstein,1781, ,1850
Märchenwald,1793,X,1930
Fuchsbau,1811, ,2010
Torfmoor,1817, ,2120
Gartenschau,1825, ,2260
Märchenwald,1840, ,2340
Eselsbrücke,1855, ,2420
Heimatmuseum,1863, ,2490
Eselsbrücke,1877, ,2590
Reiterdenkmal,1880, ,2730
Riesenrad,1881, ,2910
Hochsitz,1885, ,2980
Gartenschau,1898, ,3060
Riesenrad,1902, ,3180
Dreibannstein,1911,X,3310
Olympisches Dorf,1924, ,3470
Haus der Zukunft,1927,X,3600
Stellwerk,1931, ,3720
Dreibannstein,1938, ,3890
Stellwerk,1942, ,4020
Labyrinth,1955, ,4230
Gauklerstadl,1961, ,4310
Planetarium,1971,X,4390
Känguruhfarm,1976, ,4440
```

```
Balzplatz,1978, ,4520
Dreibannstein,1998,X,4610
Labyrinth,2013, ,4740
CO2-Speicher,2022, ,4930
Gabelhaus,2023, ,5000
```

Ausgabe

```
We have saved a total distance of 2380 using the following path:
Gabelhaus during year 1638 with a total distance of 0 --- False
Gabelhaus during year 1699 with a total distance of 0 --- False
Hexentanzplatz during year 1703 with a total distance of 160 --- True
EselsbrÃ¼cke during year 1711 with a total distance of 280 --- False
Dreibannstein during year 1724 with a total distance of 390 --- False
Dreibannstein during year 1752 with a total distance of 390 --- False
Schmetterling during year 1760 with a total distance of 540 --- True
Dreibannstein during year 1781 with a total distance of 620 --- False
MÃ¤rchenwald during year 1793 with a total distance of 700 --- True
MÃ¤rchenwald during year 1840 with a total distance of 700 --- False
EselsbrÃ¼cke during year 1855 with a total distance of 780 --- False
EselsbrÃ¼cke during year 1877 with a total distance of 780 --- False
Reiterdenkmal during year 1880 with a total distance of 920 --- False
Riesenrad during year 1881 with a total distance of 1100 --- False
Riesenrad during year 1902 with a total distance of 1100 --- False
Dreibannstein during year 1911 with a total distance of 1230 --- True
Olympisches Dorf during year 1924 with a total distance of 1390 --- False
Haus der Zukunft during year 1927 with a total distance of 1520 --- True
Stellwerk during year 1931 with a total distance of 1640 --- False
Stellwerk during year 1942 with a total distance of 1640 --- False
Labyrinth during year 1955 with a total distance of 1850 --- False
Gauklerstadl during year 1961 with a total distance of 1930 --- False
Planetarium during year 1971 with a total distance of 2010 --- True
KÃ¤nguruhfarm during year 1976 with a total distance of 2060 --- False
Balzplatz during year 1978 with a total distance of 2140 --- False
Dreibannstein during year 1998 with a total distance of 2230 --- True
Labyrinth during year 2013 with a total distance of 2360 --- False
CO2-Speicher during year 2022 with a total distance of 2550 --- False
Gabelhaus during year 2023 with a total distance of 2620 --- False
```

Quellcode

```python
# retrieving the input

with open("input.txt", "r") as filereader:
    lines = filereader.readlines()

n = int(lines[0])
del lines[0]

# formatting the input

sights = [line.split(",") for line in lines]

for i in range(n):
    sights[i][1], sights[i][3] = int(sights[i][1]), int(sights[i][3])
    if(sights[i][2] == " "):
```

```
        sights[i][2] = False
    else:
        sights[i][2] = True

result = [0]
events = {}

lastEssential = 0

indexLastSeen = {}

# short function to calculate distance between i and j using the prefix idea

def calcDistance(i,j):
    return sights[j][3] - sights[i][3]

for i in range(n):

    if(sights[i][0] in indexLastSeen.keys()):
        # add to events, update indexLastSeen
        events[i] = indexLastSeen[sights[i][0]]
    indexLastSeen[sights[i][0]] = i

    if(sights[i][2] == True):

                # finished Segment - apply DP to findout which points to remove

        # dp stuff
        dp = [0 for j in range(i-lastEssential+1)]
        for j in range(lastEssential+1, i+1, 1):
            if(j in events.keys()):
                dp[j-lastEssential] = dp[events[j]-lastEssential] + calcDistance(events[j], j)
            dp[j-lastEssential] = max(dp[j-lastEssential],dp[j-lastEssential-1])

        print(dp)

                # backtracking to know which nodes were skipped in the dp solution

        usedNodes = []
        cur = i
        while(cur > lastEssential):
            usedNodes.append(cur)
            if(dp[cur-lastEssential] == dp[cur-1-lastEssential]):
                cur -= 1
            else:
                cur = events[cur]

                # add it, but reversed

        for j in range(len(usedNodes)-1,-1,-1):
            result.append(usedNodes[j])

                # setup the variables for the next segment, make sure the last node is direclty added to
that segment

        events = {}
        lastEssential = i
        indexLastSeen = {
            sights[i][0] : i
        }

# reapply the process, we might have ended the for loop without an essential node, so our segment needs to
automatically end and be processed aswell
```

```
dp = [0 for j in range(i-lastEssential+1)]
for j in range(lastEssential+1, i+1, 1):
    if(j in events.keys()):
        dp[j-lastEssential] = dp[events[j]-lastEssential] + calcDistance(j, events[j])
    dp[j-lastEssential] = max(dp[j-lastEssential],dp[j-lastEssential-1])
# backtracking to know which nodes were skipped in the dp solution
usedNodes = []
cur = i
while(cur > lastEssential):
    usedNodes.append(cur)
    if(dp[cur-lastEssential] == dp[cur-1-lastEssential]):
        cur -= 1
    else:
        cur = events[cur]

for j in range(len(usedNodes)-1,-1,-1):
    result.append(usedNodes[j])



# calculate the total saved distance and prepare for the output

distanceSaved = 0

newSights = []

for i in range(len(result)-1):
    newSights.append([sights[result[i]][0], sights[result[i]][1], sights[result[i]][2], sights[result[i]]
[3]-distanceSaved])

    if(result[i]+1 != result[i+1]):
        # hier haben wir einen cycle gespart
        distanceSaved += sights[result[i+1]][3] - sights[result[i]][3]

newSights.append([sights[result[len(result)-1]][0], sights[result[len(result)-1]][1],
sights[result[len(result)-1]][2], sights[result[len(result)-1]][3]-distanceSaved])

# handle output

print(f"We have saved a total distance of {distanceSaved} using the following path:")

for i in range(len(newSights)):
    print(f"{newSights[i][0]} during year {newSights[i][1]} with a total distance of {newSights[i][3]} ---
{newSights[i][2]}")
```