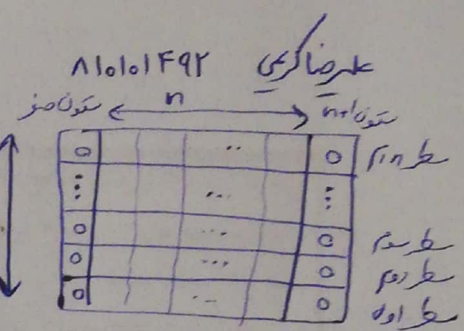
DA – HW2   ۸۱۰۱۰۱۴۹۲   علیرضا کریمی

1) الف)



تعریف: $dp[i][j]$ = بیشترین میزان کشش تا خانه در سطر i و ستون j ام

$0 \le j \le n+1$ و $1 \le i \le n$

پایه: $dp[1][j] = a_{1,j}$; $dp[k][0] = 0$; $dp[k][n+1] = 0$

ردیف اول ← $1 \le j \le n$

سطرها از پایین به بالا نام گذاری شوند

آپدیت: $dp[i][j] = \max(dp[i-1][j-1], dp[i-1][j], dp[i-1][j+1]) + a_{i,j}$

$2 \le i \le n$; $1 \le j \le n$

جواب: $\max_{1 \le j \le n} (dp[n][j])$

پیچیدگی: برای تک تک خانه های جدول $n \times (n+2)$ ، به اندازه $O(1)$ خرج آپدیت و set کردن val است. پس اردر آن $O(n^2)$ است

حافظه: درصورتی که از این جدول گفته شده استفاده کنیم ← اردر حافظه $O(n^2)$

ب)

چون داریم سطر به سطر از پایین به بالا می رویم و هر سطر تنها به سطر قبلی خود وابسته است، می توانیم فقط ردیف قبل و ردیف حال رو نگه داریم که می شه دو ردیف n تایی ← اردر حافظه $O(n)$

---

2) تعریف: $dp[i][j]$ = در روز i ام به چند حالت مختلف به مکان j رسیده

$0 \le i \le n$ ← $-M \le j \le M$

پایه: $dp[0][j] = 0$; $dp[0][0] = 1$ → چون قبل از حرکت کردن (در روز صفرم) در خانه $j = 0$ قرار داره

$-M \le j \le M$, $j \ne 0$

$1 \le i \le n$

آپدیت: $dp[i][j] = dp[i-1][j+m_j] + dp[i-1][j-m_j]$

روز i ام ← $-M \le j \le M$ ← روز i-1 ام

اگر $j+m_j \ge M$ ← $dp[i-1][j+m_j]$ را صفر در نظر می گیریم. اگر هم $j-m_j \le M$ ← $dp[i-1][j-m_j]$ را صفر در نظر می گیریم

ادامه در صفحه بعد

(ادامه سوال 2) جواب : $dp[n][t]$

همانطور که دیدید ما یک حلقه از $1 \leq i \leq n$ می‌خواهیم و یک حلقه از $-M \leq j \leq M$ و پیچیدگی آپدیت : پیچیدگی کردن هر $dp[i][j]$ از اردر $O(1)$ است ⟵ اردر پیچیدگی $O(nM)$ است.

(سوال 3)

```
int calc_function (int A[ ], int n) {        // A[i] → نفر i ام طرفدار کدام عضو انجمن است؟
                  (سایز آرایه)
    int ACM[m]      (سایز m)      // ACM[j] → یعنی درجه گمنامی عضو j ام انجمن   1 ≤ j ≤ m
    int last[m]                   // last[j] → آخرین ایندکس دیده شدن عضو j ام انجمن   1 ≤ j ≤ m
    ACM[j] = 0   for j=1 to m     → مقدار دهی اولیه درجه گمنامی همه به صفر بعدا قراره آپدیت شه
    last[j] = -1   for j=1 to m   → آخرین ایندکس دیده شدن همه افراد رو -1 می‌ذاریم اولش چون هنوز ندیدیم هیچ کدوم رو
    for i=1 to n
        ACM[A[i]] = max(ACM[A[i]], i - last[A[i]])
                    (i - last[A[i]]: اختلاف ایندکس الان که عضو A[i] رو دیدیم با آخرین باری که عضو A[i] رو دیدیم می‌تونه درجه گمنامی بزرگتر بده)
        last[A[i]] = i            ← آپدیت آخرین باری که A[i] را دیدیم به ایندکس همین الان
    end
    for j=1 to m                  // بررسی کردن بازه آخر برای همه
        ACM[j] = max(ACM[j], (n+1) - last[j])
                    ((n+1) - last[j]: از آخرین باری که دیدیم دیگه تا آخر ندیدیمش که خود این می‌تونه درجه گمنامی رو آپدیت کنه)
    end
    return min(ACM[j] for j=1 to m)      برگرداندن کمترین درجه گمنامی اعضا
}
```

4)

برای هر آزمون تمام جایگشت‌های ممکن ($3!$ حالت) را می‌سازیم $\Longleftarrow$ به نوعی $6N$ آزمون داریم.

(از آنجا که درصد شیمی و فیزیک باید اکیدا صعودی باشند، نمی‌توانیم از یک حالت یک آزمون دو تا برداریم)

$6N$ آزمون را طبق فیزیک sort می‌کنیم $\xleftarrow{\text{اردر}}$ $O(N\log N)$ (می‌توانستیم طبق شیمی این کار را بکنیم) و در A ذخیره می‌کنیم

فرض کنید داریم:

آرایه A : آزمون‌های sort شده طبق فیزیک

آرایه B : " " " " شیمی

آزمون‌های sort شده طبق فیزیک را در A داریم. دنبال یک زیر دنباله اکیدا صعودی بین ترازهای شیمی آزمون‌ها هستیم که حداکثر مجموع ترازهای ریاضی را بدهند. فرض کنید که sort شده آزمون‌ها طبق شیمی را در B داشته باشیم.

برای راحتی کار تعریف می‌کنیم یک map که اندیس یک آزمون در A را map کند به اندیس همان آزمون در B. درواقع

$A[i]$ ← آزمون iام طبق sort فیزیک $B[j]$ آزمون jام طبق sort شیمی و $map[i]=j$
if $A[i]=B[j]$

(اردر پیدا کردن و حساب کردن این map از اردر حداکثر $O(N^2)$ است، پس شرایط مسئله را نقض نمی‌کند.)

یک آرایه dp تعریف می‌کنیم با $6N$ عضو و داریم: تعریف: $dp[i] = 0 \quad 1 \leqslant i \leqslant 6N$

$dp[i]$ درواقع حداکثر مجموع درصد ریاضی تا آزمون iام است

for $i = 1$ to $6N$

$$dp(map[i]) = \underbrace{\max\big(dp(map[i]-k)\big)}_{\text{حداکثر مجموع درصد ریاضی‌ها را با درصد شیمی کمتر}} + A[i].m$$

درصد ریاضی آزمون iام در آرایه A

for $1 \leqslant k < map[i]$

end

اندیس آزمون iام در آرایه B که طبق درصد شیمی sort شده بود.

return $\max(dp(k))$ $\quad 1 \leqslant k \leqslant 6N$ → حداکثر مجموع درصد ریاضی را برمی‌گردانیم با max گرفتن بین کل مقادیر

توضیح الگوریتم: فرض کنید dp به این شکل است:

$$dp \quad \begin{array}{|c|c|c|c|c|} \overset{1}{0} & \overset{2}{0} & \overset{3}{0} & \cdots & \overset{6N}{\ } \end{array}$$

$$C_1 < C_2 < C_3 < \cdots < C_{6N}$$

حال روی درصدهای شیمی آزمون‌ها طبق مرتب شده نسبت به فیزیک (آرایه A) حلقه می‌زنیم. آزمون iام دارای درصد شیمی C_j است. تمامی مقادیر dp از 1 تا $j-1$ که مربوط به مقادیر C_1 تا C_{j-1} هستند، اگر غیرصفر باشند، یعنی قبل از رسیدن به C_j در آرایه A آنها را دیده‌ام. پس با اضافه کردن تراز ریاضی مربوط به C_j به مجموع ترازهای ریاضی ذخیره شده در $dp[1]$ تا $dp[j-1]$ درواقع C_j (آزمون با تراز) را به دنباله اکیدا صعودی خود اضافه کردیم و اینگونه مطمئنیم یک زیر دنباله اکیدا صعودی داریم که با C_j پایان یافته. که اگر غیرصفر باشند یعنی آنها را قبلا دیده‌ام

(ادامه ۴

محاسبه پیچیدگی: به ازای هر آزمون k در آرایه B، $k-1$ مقدار را مقایسه کرده و آپدیت را انجام می‌دهیم. پس به ازای هر $1 \leq k \leq 6N$ این کار را انجام می‌دهیم و داریم: $1+2+\cdots+6N-1 = \frac{6N(6N-1)}{2}$

$\Longrightarrow$ پیچیدگی آپدیت $= O(N^2)$

از آنجایی که در مراحل قبلی هم $O(N\log N)$ برای sort و $O(N^2)$ برای تابع map داریم، پس پیچیدگی کل از اردر $O(N^2)$ است.

---

5)

تعریف: $dp[i,j] =$ حداکثر میزان نمره لازم برای خرج کردن برای بازه i تا j

$1 \leq i \leq j \leq n$

پایه: $dp[i,i]=0$ ؛ $dp[i,i+1] = i$ ؛ $dp[i,i+2] = i+1$

$1 \leq i \leq n$ ، $1 \leq i < n$ ، $1 \leq i < n-1$

توضیح: وقتی به یک عدد برسیم (یعنی $dp[i,i]$) قطعا همان جواب است و نیاز به خرج بیشتر نیست. درصورتی که به دوتا عدد متوالی رسیدیم عدد کوچک‌تر را انتخاب می‌کنیم که درصورت اشتباه بودن و خرج آن، مطمئن می‌شویم عدد بزرگتر جواب است و دیگر خرج نمی‌کنیم.
اگر بازه ۳ تایی بود، عدد وسط را انتخاب می‌کنیم. اگر غلط بود، با گفتن چپ و راست توسط دکتر دوستی، به جواب درست و قطعی می‌رسیم

آپدیت: $dp[i,j] = \min\left(k + \max(dp[i,k-1], dp[k+1,j])\right)$

$1 \leq i < j \leq n$ ، $i < k < j$

(البته الان که فکر می‌کنم، حالت $dp[i,i+2]$ را هم می‌توانستیم در آپدیت بیاوریم و نیازی به جدا کردن به عنوان حالت پایه نبود)

جواب: $dp[1,n]$

پیچیدگی: مثل مسئله ضرب ماتریس‌هاست که یک جدول $n \times n$ داریم و باید ~~آن~~ نیمه بالا راست آن را به صورت مورب پر کنیم و برای هر خانه باید $j-i-2$ مقایسه (از اردر $O(n)$) انجام دهیم $\Leftarrow$ پیچیدگی از اردر $O(n^3)$ خواهد بود.

6)

تعریف dp : $dp[i] =$ تعداد طلسم های لازم برای پیمودن فاصله i

$0 \le i \le T$

پایه : $dp[0] = 0$

آپدیت : $dp[t] = x$ در صورتی که $t = 2^x - 1$ باشد داریم

$0 < t \le T$

در غیر اینصورت، آپدیت به این صورت است:

$$dp[t] = \min \begin{cases} dp[2^{\lceil \log(t+1) \rceil} - 1 - t] + \lceil \log(t+1) \rceil + 1 \longrightarrow \text{برای تغییر جهت} \\ dp[t - (2^i - 1)] + i + 2 \longrightarrow \text{دو بار تغییر جهت برای reset کردن حرکت در همین جهت} \end{cases}$$

$1 \le i \le \lfloor \log_2(t+1) \rfloor$

حالت اول زمانی رخ می‌دهد که انقدر در همین جهت می‌رویم که جایگاه T را رد می‌کنیم. بلافاصله بعد از اینکه T را رد کردیم، تغییر جهت داده و به سمت آن برمی‌گردیم و dp فاصله تا T را حساب می‌کنیم. $\lceil \log(T+1) \rceil$ تعداد طلسم های لازم برای رد شدن از T برای اولین بار است.

حالت دوم زمانی است که ما یک تعداد پرش انجام می‌دهیم مثلا i تا پرش می‌کنیم و $2^i - 1$ از فاصله را طی می‌کنیم. حال حرکت خود را با خوردن دو طلسم B، reset می‌کنیم و dp فاصله باقی مانده تا T را حساب می‌کنیم. بین حالت ۱ و ۲، min می‌گیریم.

جواب : $dp[T]$ (بدیهی است که $2^{\lceil \log t+1 \rceil} - 1$ از دو برابر t کمتر است)

پیچیدگی : از آنجایی که برای هر فاصله توانستیم که به dp با فاصله کمتر برسیم این آپدیت را بگونه‌ای بنویسیم که به زیرمسئله های کوچک‌تر برسیم و برای هر آپدیت حداکثر $\log t$ تا هزینه دادیم که $0 < t \le T$. پس هزینه آپدیت حداکثر $O(\log T)$ است. از آنجا که T فاصله مختلف داریم و برای هر کدام این آپدیت را داریم، در کل باید به اندازه $O(T \log T)$ هزینه دهیم.