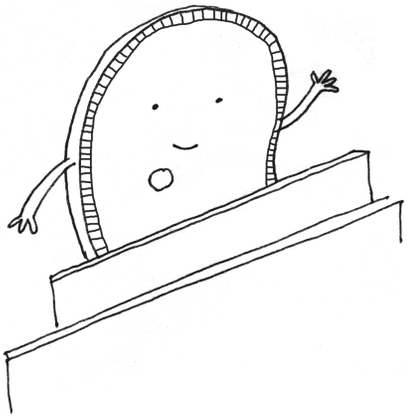
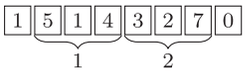




Dla $k = 2$ oraz tablicy

$$a = [1, 5, 1, 4, 3, 2, 7, 0]$$

poprawnym wynikiem jest 14. Jeden kot może pilnować fragmentu korytarza na metrach od 2 do 4 (łapiąc 6 z 10 myszy), a drugi może pilnować fragmentu na metrach od 5 do 7 (łapiąc 8 z 12 myszy).

Informatyczny kącik olimpijski (94): Myszy

Tym razem zajmiemy się zadaniem, które pojawiło się na kolokwium dla studentów pierwszego roku informatyki na Uniwersytecie Warszawskim. W korytarzu harcują myszy. Korytarz ma długość n metrów. Dana jest tablica n nieujemnych liczb całkowitych $a[\,]$ opisująca, gdzie jest ile myszy: dla $1 \leqslant i \leqslant n$ na i-tym metrze korytarza (patrząc od strony wejścia) jest $a[i]$ myszy. Dysponujesz k kotami ($k \leqslant n$). Twoim zadaniem jest takie rozmieszczenie kotów w korytarzu, żeby złapały jak najwięcej myszy. Każdy kot może pilnować ustalonego przez Ciebie spójnego fragmentu korytarza (na przykład, może mieć założoną smycz odpowiedniej długości, przymocowaną do podłogi pośrodku pilnowanego fragmentu korytarza). Fragmenty korytarza pilnowane przez różne koty nie mogą zachodzić na siebie (żeby koty się nie pobiły, a smycze nie poplątały), choć mogą się stykać. Niektóre fragmenty korytarza mogą pozostać niepilnowane przez żadnego kota. Kot, który pilnuje fragmentu od i-tego do j-tego metra włącznie (dla $i \leqslant j$), na którym znajduje się $s = a[i] + a[i+1] + \cdots + a[j]$ myszy, złapie: $\max(s - (j-i)^2, 0)$ myszy. Należy wyznaczyć maksymalną liczbę myszy, jakie mogą złapać koty (patrz rysunek).

Obojętnie, jak zabierzemy się za to zadanie, przyda się nam tablica sum prefiksowych: $s[j] = \sum_{1 \leqslant i \leqslant j} a[i]$. Wtedy liczba myszy na fragmencie $[i, j]$ przedstawia się jako $s[j] - s[i-1]$.

Omawiane zadanie to zadanie na programowanie dynamiczne. Niech $d[j, l]$ oznacza maksymalną liczbę myszy, jakie może złapać l kotów na pierwszych j metrach korytarza. Ponieważ niektóre fragmenty mogą być niepilnowane, więc rekurencję budujemy, rozpatrując dwa przypadki: albo j-ty metr nie jest pilnowany, albo jest i l-ty kot pilnuje fragmentu $[i, j]$ (od i-tego do j-tego metra włącznie):

$$(*) \qquad d[j,l] = \max\Big(d[j-1,l], \max_{0 \leqslant i \leqslant j}\big(d[i-1,l-1] + s[j] - s[i-1] - (j-i)^2\big)\Big).$$

Podstawowa implementacja zadziała w czasie $O(n^2 k)$. Takie było oczekiwane rozwiązanie na kolokwium, niemniej jednak okazało się, że można to zrobić lepiej (ach, ci studenci). Aby przyspieszyć rozwiązanie, należy w odpowiedniej kolejności wypełniać tablicę d. Główna pętla powinna przebiegać po l. Ustalmy l i załóżmy, że mamy obliczone wartości $d[j, l-1]$ dla wszystkich j. Będziemy teraz obliczać element $d[j, l]$. Kluczem do usprawnienia rozwiązania jest porównanie możliwych dwóch ustawień l-tego kota, mianowicie gdy patroluje on fragment $[i_1, j]$ lub fragment $[i_2, j]$, gdzie $i_1 < i_2$. Drugie ustawienie jest lepsze od pierwszego, jeśli:

$$d[i_1-1, l-1] + s[j] - s[i_1-1] - (j-i_1)^2 < d[i_2-1, l-1] + s[j] - s[i_2-1] - (j-i_2)^2.$$

Po przekształceniu nierówność przyjmie postać:

$$(**) \qquad j > \big(d[i_1-1,l-1] - d[i_2-1,l-1] - s[i_1-1] + s[i_2-1] - {i_1}^2 + {i_2}^2\big) / \big(2(i_2 - i_1)\big).$$

Prawa strona nierówności $(**)$ zależy tylko od i_1 i i_2 (oraz od l); oznaczmy ją przez $p(i_1, i_2)$. Widzimy, że pierwsze ustawienie jest lepsze dla j mniejszych od $p(i_1, i_2)$, po czym, począwszy od $\lceil p(i_1, i_2) \rceil$, lepsze staje się drugie ustawienie. Ta obserwacja umożliwia przeglądanie możliwych ustawień l-tego kota w sposób bardziej uporządkowany tak, że na bieżąco eliminujemy takie ustawienia, które są już od pewnych j gorsze. Trzymamy kolejkę dwukierunkową początków fragmentów $1 \leqslant i_1 < \cdots < i_m < j$, taką że

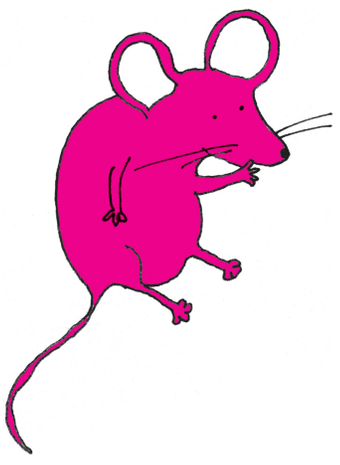
$$j < \lceil p(i_1, i_2) \rceil < \lceil p(i_2, i_3) \rceil < \cdots < \lceil p(i_{m-1}, i_m) \rceil.$$

Innymi słowy, dla odpowiednio małych j najlepiej przydzielać l-temu kotu fragment $[i_1, j]$, gdy j osiągnie $\lceil p(i_1, i_2) \rceil$ – fragment $[i_2, j]$, i tak dalej aż do $j \geqslant \lceil p(i_{m-1}, i_m) \rceil$, gdzie optymalny przydział to $[i_m, j]$. Przetwarzanie j-tego metra korytarza składa się z następujących kroków:

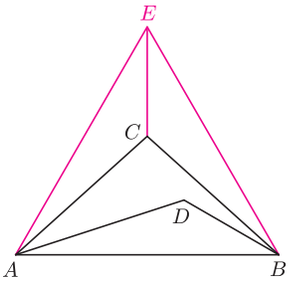
1. Jeżeli kolejka ma co najmniej dwa elementy oraz $j \geqslant \lceil p(i_1, i_2) \rceil$, to usuwamy z przodu kolejki i_1, gdyż od tego momentu lepsze będą fragmenty o początku i_2.
2. Jeżeli kolejka ma wciąż co najmniej dwa elementy, to sprawdzamy, czy $\lceil p(i_{m-1}, i_m) \rceil \geqslant \lceil p(i_m, j) \rceil$. Jeżeli tak, to fragmenty o początku i_m są zdominowane przez fragmenty o początkach i_{m-1} lub j, usuwamy więc i_m z końca kolejki i powtarzamy krok 2.
3. Dodajemy j na koniec kolejki.
4. Obliczamy $d[j, l]$, korzystając ze wzoru $(*)$, ale możemy to już teraz zrobić w czasie $O(1)$, gdyż wiemy, że maksimum jest osiągane przy $i = i_1$, gdzie i_1 jest aktualnym elementem z przodu kolejki.

W ten sposób otrzymaliśmy rozwiązanie o złożoności czasowej $O(nk)$.

Jakub PAWLEWICZ



Rozwiązanie zadania M 1497.



Zbudujmy taki trójkąt równoboczny ABE, że punkt C leży w jego wnętrzu. Wówczas punkty C i E leżą na symetralnej odcinka AB, więc $\sphericalangle AEC = 30^\circ$. Ponadto

$$\sphericalangle CAE = 60^\circ - \sphericalangle CAB = \\ = 60^\circ - \frac{1}{2}(180^\circ - \sphericalangle ACB) = 18^\circ.$$

W takim razie trójkąty ADB i ACE są przystające, w szczególności $AC = AD$. Stąd łatwo otrzymujemy

$$\sphericalangle ACD = \frac{1}{2}(180^\circ - \sphericalangle CAD) = 78^\circ.$$