Riešenie 3. úlohy kategórie A

Jozef Komáromy

November 11, 2020

1 Súťažná úloha

Tento program je zapísaný v jazyku Python.

1.1 Inicializácia premenných

Zadané údaje

n, k a a sú konštanty - počas behu programu sa nemenia. Konštanta a je list, ktorého prvkami sú šírky stánkov $a_1, a_2 ... a_n$. V programe zodpovedá prvému stánku index 0.

```
n = 7
k = 600
a = [200, 100, 200, 300, 200, 200, 100]
```

Premenné pre uloženie výsledných hodnôt

```
vysledok_d = 0
vysledok_l = 0
```

1.2 Hlavný algoritmus

Tento algoritmus pre každý stánok nájde počet stánkov ktoré celtou možno zakryť ak zakryjeme daný stánok ako prvý. Napriek tomu, že tento proces vykonávame pre každý jeden stánok, algoritmus používa výsledky platné pre predchádzajuci stánok ako základ pre ďalšie výpočty.

Algoritmus pracuje s rozmedzím stánkov, ktoré určujú premenné l a r. Premenná l je indexom prvého stánku tohto rozmedzia a premenná r indexom posledného. Súčet šírok stánkov v tomto rozmedzí, dĺžka celty potrebnej na zakrytie tohto rozmedzia, je uložený v premennej p.

Počiatočné hodnoty týchto premenných zahŕňajú všetky stánky. Preto $l = 0$ je prvý stánok a $r = n - 1$ je posledný stánok, ich celková šírka bude súčet všetkých prvkov listu a ($p = sum(a)$).

```
l = 0
r = n - 1
p = sum(a)
```

Ak je celta väčšia ako všetky stánky spolu ($k >= p$), jediným riešením bude, že zakryjeme všetky stánky ($vysledok_d = n$) a začneme od prvého ($vysledok_l = 0$).

```
if p < k:
  vysledok_d = n
  vysledok_l = 0
```

Naopak, ak celta nepostačuje ani na zakrytie najmenšieho stánku ($k < min(a)$), riešením bude že sa nedá zakryť žiadny stánok (program vypíše 0̈ 0̈).

```
elif k < min(a):
  vysledok_d = 0
  vysledok_l = 0
```

Algoritmus pre každý stánok l od prvého po posledný nájde stánok r, ktorý sa ako posledný dá zakryť celtou natiahnutou od stánku l.

```
else:
  while l <= n:
```

Pokiaľ je rozmedzie menšie ako celta ($p < k$), pridávame na pravú stranu rozmedzia ďalší stánok zväčšením indexu r o 1 ($r+ = 1$) a k veľkosti rozmedzia pridáme veľkosť novo pridaného stánku ($a[r]$). Podmienka ($r + 1 < n$) bráni tomu aby v niektorých prípadoch prekročilo rozmedzie za posledný stánok.

```
    while p < k and r + 1 < n:
      r += 1
      p += a[r]
```

Pokiaľ je rozmedzie väčšie ako celta ($p > k$), odoberáme z pravej strany rozmedzia posledný stánok odčítaním veľkosti posledného stánku od veľkosti rozmedzia ($r- = 1$) a zmenšením indexu r o 1 ($r- = 1$). Podmienka ($l+1 < r$) bráni tomu aby sa vo výnimočných prípadoch pravá hranica rozmedzia dostala pred ľavú.

```
    while p > k and l + 1 < r:
      p -= a[r]
      r -= 1
```

Vypočítame d, počet stánkov v rozmedzí.

```
    d = r - l
```

Skontrolujeme či celta skutočne dokáže zakryť rozmedzie podmienkou $k >= p$ a či nové riešenie d je lepšie ako posledné najlepšie ($d > vysledok_d$). Ak áno, nastavýme premenné výsledkov na nové riešenie.

```
    if k >= p and d > vysledok_d:
      vysledok_d = d
      vysledok_l = l
```

Z ľavej strany rozmedzia odoberieme prvý stánok: odčítame veľkosť prvého stánku rozmedzia ($p- = a[l]$) a ľavú hranicu l posunieme o 0 ($l+ = 1$). Po tomto kroku sa cyklus opakuje pre ďalší stánok až po posledný.

```
    p -= a[l]
    l += 1
```

1.3 Vypísanie riešenia

Vypíšeme výsledok. K výslednej hodnote l pridáme 0, pretože je indexom pythonového listu (začína od 0).

```
print(vysledok_d, vysledok_l + 1)
```