Feladatleírás

A bevásárló kosarunkban n árucikk található. Fizetéskor egyesével olvassa be az összes árucikket az eladó. Minden árucikknek adott az ára és az idő másodpercben, ameddig az eladónak tart beolvasni az adott terméket. Amíg az eladó egy árucikk árának leolvasásával foglalkozik, a kosarunkból termékeket lophatunk 1 termék/másodperc sebességgel. Mekkora a minimum összeg, amennyit ki kell fizetnünk az eladónak, ha ami határozzuk meg, milyen sorrendben olvassa le az eladó a termékek árát.

Eredeti feladat

https://codeforces.com/contest/19/problem/B

Megoldás

Hibás megoldás

Olyan sorrendben döntsük el, hogy megvesszük-e az aktuális árucikket, ahogy adva van, és amikről már döntöttünk, a további termékek vizsgálatakor nem foglalkozunk. *Gond, hogy* így nem minden esetet vizsgálunk meg. *Megoldás* lesz erre a problémára a **dinamikus programozás** (lásd lejjebb).

Helyes, de lassú megoldás

Vizsgáljuk meg az összes *lehetséges* variációt, hogy melyik termékeket vásároljuk meg, illetve melyikeket lopjuk el, majd ezen esetek minimumát keressük meg. *Gond, hogy* minden egyes esetet megvizsgálunk, így a programunk futási ideje nem lesz elfogadható.

Helyes megoldás

Ötlet

Alkalmazzuk a **dinamikus programozás** elvét. Vegyük csak alapul a *Hátizsák problémát*. Mostani feladatunk nagyban megegyezik vele, csak súly helyett idő változóval, amit az egyes termékek megvásárlása befolyásol).

Megvalósítás

Mikor beolvassuk az i-edik elemet (annak **árát** -> $\mathbf{c_i}$, valamint az **időt**, amennyibe az eladónak telik beolvasni az adott termék árát -> $\mathbf{t_i}$), mérlegeljük, hogy az addigi beolvasott termékekért kiadott pénz több-e, mint az, amennyit akkor kell fizetnünk, ha

beolvassuk a mostani terméket, valamint, hogy ezzel hány másik terméket tudunk ellopni az előzőek közül. Így a két összeg az eddigi megvásárolt termékek **ára**, valamint az aktuális termék **ára**, és az aktuális **t** értéke szabja meg. Részletes algoritmus a mellékelt kódban található.

Költségesség

Műveletigény

Mivel a dinamikus programozáshoz egy "táblázatot" használunk, annak meghatározásának idejét kell vizsgálni. Minden részprobléma (adott elemek kiszámítása) konstans idejű, így a futási idő n-nel arányos.

Tárhely

A feladat megoldásához egy egydimenziós tömbre, valamint az aktuális értékek eltárolására kettő egész számnak, n eltárolására további egy egész számnak kell területet lefoglalni. Az egydimenziós tömb hossza n+1.

Implementáció

Pyhton 3.7

```
n = int(input(""))
        dp = []

        for i in range(0, n+1):
                dp.append(maxNum)

        dp[0] = 0

        for i in range(0, n):
                tc = input("").split(' ')
                t = int(tc[0])
                c = int(tc[1])

                for j in range(n, -1, -1):
                        if (j+t+1 < n):
                                dp[j+t+1] = min(dp[j+t+1], dp[j] +
c)
                        else:
                                dp[n] = min(dp[n], dp[j] + c)

        print(dp[n])
```