Projects/ Projektek lépései

Ez a program egy dinamikus programozás (DP) alapú megoldást használ, hogy megtalálja a maximális profitot, amelyet egy sor projektek közül választhatunk ki úgy, hogy egy projekt sem ütközik egy másikkal. A feladatban adott n projekt, mindegyik rendelkezik egy kezdő- és befejező dátummal, valamint egy jutalommal. Az a cél, hogy úgy válasszunk projekteket, hogy azok nem fedjék egymást, és a lehető legnagyobb jutalmat szerezzük.

A program lépésről lépésre történő magyarázata:

1. Projektek rendezése befejezésük szerint

projects.sort(key=lambda x: x[1])

A projektek listáját befejező napjaik szerint rendezzük. Ez azért fontos, mert ha egy projekt befejezése előtt nem kezdünk új projektet, akkor az előző projektek befejezésének sorrendjében végig tudunk haladni, és könnyebben meghatározhatjuk, hogy a következő projektet ki tudjuk-e választani. A lambda x: x[1] azt mondja a sort függvénynek, hogy a projekt második eleme (a befejező nap) alapján rendezzük.

2. DP tárolása

dp = [0] * (n + 1)

A dp[i] egy dinamikus programozás lista, amely tárolja a maximális profitot, ha az első i projektet választjuk ki. A dp[0] értéke 0, mert ha nem választunk semmilyen projektet, akkor a jutalom 0. A lista n + 1 hosszú, hogy az indexelés egyszerű legyen.

3. Segédlista a befejező napok gyors kereséséhez

ends = [p[1] for p in projects] # Befejező napok listája

A ends lista tárolja az összes projekt befejező napját. Erre a listára szükség lesz, amikor azt kell keresnünk, hogy egy projekt nem ütközik-e más projekttel. Ez a lista segít gyorsan megtalálni azt a projektet, amelynek a befejező napja a legközelebb van ahhoz a naphoz, amikor az aktuális projektet elkezdjük.

4. Ciklus a projekteken

```
for i in range(1, n + 1):
  start, end, reward = projects[i - 1]

  # Ne válasszuk ezt a projektet, akkor a jutalom ugyanaz, mint az előzőé
  dp[i] = dp[i - 1]
```

Bukodi-Vass Annamária (D9T3LQ)
Digitális kultúra tanára

A ciklusban végigiterálunk a projekteken. Az aktuális projekt kezdő- és befejező napja és jutalma (start, end, reward) alapján dolgozunk.

- **dp[i] = dp[i - 1]**: Az alapértelmezett eset az, hogy az aktuális projektet nem választjuk, így a maximális profit ugyanaz, mint az előző projekt választásánál. Ez az alapérték.

5. A legutolsó ütközésmentes projekt keresése

idx = bisect.bisect_right(ends, start - 1)

Ez a sor kulcsfontosságú a dinamikus programozás sebességében, mivel segít gyorsan megtalálni, hogy az aktuális projekt előtt melyik projekt fejeződik be legkésőbb. Az ends lista tartalmazza a projektek befejező napjait, és a bisect_right függvény segítségével megkeressük a legnagyobb indexet, amelynél az érték kisebb, mint a jelenlegi projekt kezdőnapja (start - 1).

- **Miért start - 1?** Azért, hogy csak azokat a projekteket vegyük figyelembe, amelyek ténylegesen befejeződtek a jelenlegi projekt kezdete előtt. Ha a befejező napuk kisebb, mint a kezdő nap, akkor azok nem ütköznek a jelenlegi projekttel.

Ezután idx értéke az lesz, hogy melyik projekt utolsó olyan projekt, amely nem ütközik az aktuális projekttel.

6. Frissítjük a DP értéket

dp[i] = max(dp[i], dp[idx] + reward)

Miután megtaláltuk a legutolsó ütközésmentes projektet (idx), frissítjük a dp[i] értéket. Ha a dp[idx] az előző legnagyobb profit, akkor az aktuális projekt jutalmát hozzáadjuk ehhez a profithoz, hogy meghatározzuk, mi lenne a maximális profit, ha az aktuális projektet is választjuk.

7. Eredmény visszaadása

return dp[n]

A ciklus végén a dp[n] tárolja a maximális profitot, amit a n projekt kiválasztásával érhetünk el, és ezt visszaadjuk. Az n index azt jelenti, hogy figyelembe vettük az összes projektet.

8. Bemenet és kimenet

A fő programrészben beolvassuk a bemenetet, amely az összes projekt kezdő-, befejező napját és a jutalmakat tartalmazza. Ezt egy listába (projects) gyűjtjük. Végül meghívjuk a max_profit függvényt, és kiírjuk az eredményt.

```
n = int(input())  # A projektek száma

projects = []
```

```
for _ in range(n):
    a, b, p = map(int, input().split())  # a: kezdő nap, b: befejező nap, p: jutalom
    projects.append((a, b, p))

print(max_profit(n, projects))
```

Összegzés

Ez a program a dinamikus programozás és a bináris keresés (segítségével a bisect_right-t) kombinálásával oldja meg a problémát. A program hatékonyan meghatározza a maximális profitot a projektek kiválasztásakor, figyelembe véve, hogy egy projektet csak akkor választhatunk, ha nem ütközik más projekttel. A bináris keresés használata jelentősen csökkenti a futási időt, így a program képes nagyobb adatállományok kezelésére is.

A futási idő PyPy3-ban tökéletes lett, a másikban (CPython3) időtúllépést jelzett néhány tesztnél.

CSES - Projects - Results

Submission details

Task:	Projects
Sender:	D9T3LQ
Submission time:	2024-11-16 19:37:45 +0200
Language:	Python3 (PyPy3)
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.04 s	»
#2	ACCEPTED	0.04 s	»
#3	ACCEPTED	0.04 s	»
#4	ACCEPTED	0.04 s	»
#5	ACCEPTED	0.04 s	»
#6	ACCEPTED	0.83 s	»
#7	ACCEPTED	0.84 s	»
#8	ACCEPTED	0.85 s	»
#9	ACCEPTED	0.84 s	»
#10	ACCEPTED	0.86 s	»
#11	ACCEPTED	0.56 s	»
#12	ACCEPTED	0.55 s	»
#13	ACCEPTED	0.04 s	»
#14	ACCEPTED	0.04 s	»