Řešení

Úloha je poměrně jednoznačná, takže se pustím rovnou do řešení. Budu uvažovat o hmotnosti a zajímavosti pytlíku jako o dvou různých polích, kde stejné indexy budou odkazovat na vlastnosti téhož pytlíku. Toto nezmění prostorovou komplexitu a přijde mi to lepší pro znázornění v pseudokódu.

Tato úloha je dost podobná klasické úloze o maximálním součtu. Jen, místo komplikace se zápornými čísly tu máme komplikaci s omezenou délkou úseku. Ze zadání je zřejmé, že jestliže máme nějaký úsek provazu a naše síla nám umožní vzít ještě jeden pytlík, vezmeme ho. Tedy, nemusíme řešit všechny možné úseky a jejich zajímavosti, ale jen všechny nejdelší možné úseky, které jsme schopni unést a najít nejzajímavější z nich.

Jak tyto nejdelší možné úseky najdeme? Velmi jednoduše, začneme z prvního pytlíku a nabereme další pytíky, dokud je už neudržíme. Takhle získáme první úsek. Abychom mohli nabrat další pytlík, musíme se zbavit nějakých, které už máme. Protože pracujeme v souvislých úsecích, musíme opusit první pytlík, který jsme nabrali. Jestli už můžeme nabrat další pytlík, tak můžeme v algoritmu pokračovat dál. V případě, že ne, tak opustíme opět první pytlík z úseku. Toto můžeme zopakovat, dokud nenabereme poslední pytlík na celém provazu, což znamená, že jsme získali poslední nejdelší možný úsek.

Nejzajímavější z těchto úseků najdeme jednoduše při porovnání pokaždé, když nabereme nový pytlík. Podrobnější ukázka algoritmu si ukážeme ve formě pseudokódu:

Algorithm 1 $largestLimitedSum(H, Z, M)$

Input: array H of weights, array Z of interests, maximum weight M

```
i ← 0
j ← 0
curInterest ← Z_0
maxInterest ← Z_0
curWeight ← H_0
bestSumInd ← (i, j)
while j < len(H) do
    j += 1
    curWeight += H_j
    curInterest += Z_j
    while curWeight > M do
        curWeight -= H_i
        curInterest -= Z_i
        i += 1
    end while
    if curInterest > maxInterest then
        maxInterest ← curInterest
        bestSumInd ← (i, j)
    end if
end while
return bestSumInd
```

Output: pair $bestSumInd$, containing the start and end of the best sum

Pseudokód byl napsaný s předpokladem, že vstup je už uložený. V takovém případě nemůžeme dosáhout lépší prostorové složitosti než $\mathcal{O}(n)$, čehož jsme dosáhli. I kdybychom neměli vstup uložený a

museli bychom ho ukládat, tak bychom museli jenom ukládat úsek, který nás zrovna zajímá. Tento způsob by sice zabral trochu méně prostoru a zároveň by celková komplexita zůstala lineární. Jinak jsme také dosáhli časové složitosti $\mathcal{O}(n)$, protože na každý pytlík se koukneme maximálně dvakrát — jednou při posunu j a podruhé při posunu i. Lepší než lineární časovou složitost také nedokážeme, protože přeci jen musíme alespoň jednou ověřit zajímavost každého prvku.