AiSD L2

Maurycy Borkowski

17.03.2021

Zadanie 2

```
I <- lista par odcinkow
S = set()
sorted(A, key = lambda x: x.fk)
S.add(A[0])

last = A[0].k
for i in range(2,n):
    if A[i].p > last:
        S.add(A[i])
        last = A[i].k
```

Sortujemy odcinki rosnąco po ich końcach.

Bierzemy zachłannie odcinki jeżeli możemy tzn. nie nachodzą na siebie. Możemy go wziąć jeżeli początek jest większy niż koniec ostanio wziętego (ostatnio wzięty ma największy koniec w S, na początku je sortowaliśmy).

Oznaczmy jako I_0 odcinek o najmniejszym końcu. ze wszystkich odcinków i I_k odcinek o najmniejszym końcu w dowolnym rozwiązaniu optymalnym B. Zauważmy, że rozwiązanie $A = B \setminus \{I_k\} \cup \{I_0\}$ też jest optymalne bo dodany odcinek nie nachodzi się na żaden inny (B było poprawnym rozwiązaniem i $k_0 < k_k$) oraz $|A| = |B|$.

Argumentacje możemy indukcyjnie powtórzyć dla podproblemu ze zbiorem odcinków $I' = \{j \in I : p_j \geqslant k_1\}$, tam ponownie pokazujemy, że zachłanne wzięcie najmniejszego (względem końca) odcinka nie zepsuje nam optymalności rozwiązania.

Zadanie 3

```
M = a*b + 1
while aPb > 0:
    k = min([i if 1/i <= a/b for i in range(1,M)])
    aPb = aPb - 1/k
```