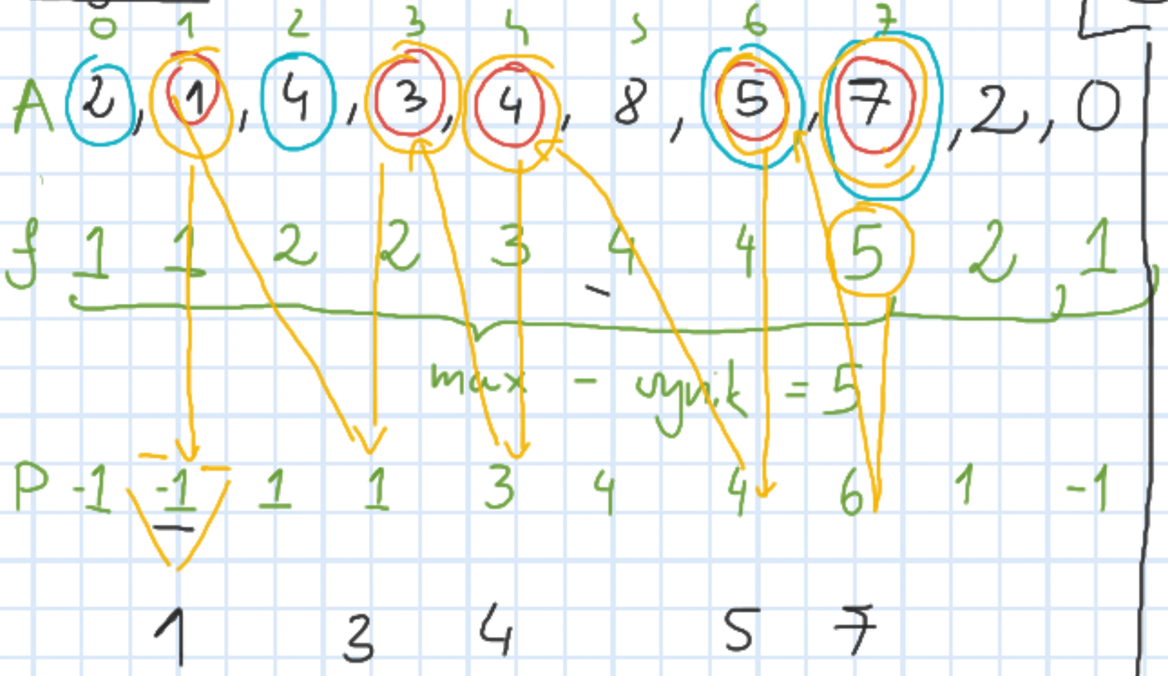
ASD - Wykład 6

Najdłuższy rosnący podciąg

Dane: $A[0,\dots,n-1]$ - tablica liczb

Zadanie: Znaleźć długość najdłuższego (niekoniecznie spójnego) rosnącego podciągu.

Przykład



① Ustalamy funkcję, którą będziemy obliczać

$f(i)$ = długość najdłuższego rosnącego podciągu w tablicy $A[0,\dots,i]$ kończącego się liczbą $A[i]$

wynik: $\max_{i \in \{0,\dots,n-1\}} f(i)$

② Wyrażenie funkcji f w postaci rekurencyjnej

$$f(i) = \max\{ f(j) + 1 \mid j < i \wedge A[j] < A[i] \}$$

$$f(0) = 1$$

konwencja: $\max \emptyset = 1$

③ implementacja

Implementacja

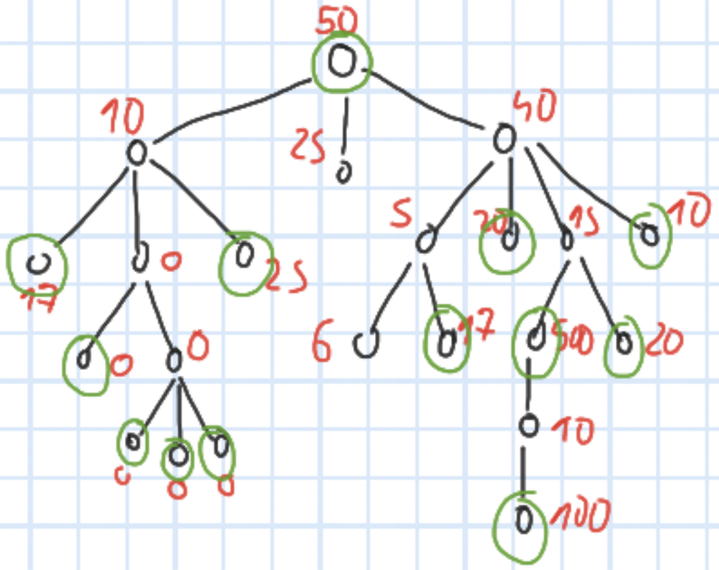
```
def lis(A):
    n = len(A) , maxi = 0
    F = [1 for i in range(n)]
    P = [-1 for i in range(n)]

    for i in range(1, n):
        for j in range(i):
            if A[i] > A[j] and F[j]+1 > F[i]:
                F[i] = F[j] + 1
                P[i] = j

        if F[i] > F[maxi]:
            maxi = i

    return maxi, F, P
```

```
def printSol(A, P, i):
    if P[i] ≠ -1:
        printSol(A, P, P[i])
    print(A[i])
```

$O(n^2)$

da się rozwiązać $O(n \log n)$

Problem imprezy firmowej



impreza jest dopuszczalna jeśli dla żadnego zaproszonego pracownika nie zaprosiliśmy jego bezpośredniego przełożonego

wartością imprezy jest suma współczynników fun zaproszonych

```
class Employee:
    def __init__(self, fun):
        self.fun = fun
        self.emp = []      <- tablica dzieci węzła
        self.f = -1
        self.g = -1
```

zadanie: znaleźć wartość najlepszej dopuszczalnej imprezy

① Określenie obliczanych funkcji

v - węzeł drzewa

f(v) - wartość najlepszej imprezy poddrzewa zakorzenionego w v

g(v) - j.w. pod warunkiem, że v nie idzie na imprezę

wynik: f(root)

② Zależności rekurencyjne

$$g(v) = \sum_{u \text{ - pracownik } v} f(u)$$

$$f(v) = \max\left(g(v), fun(v) + \sum_{u \text{ - pracownik } v} g(u) \right)$$

③ Implementacja

```
def g( v ):
  if v.g ≠ -1 : return v.g
  v.g = 0
  for u in v.emp:
    v.g += f( u )
  return v.g
```

```
def f(v):
  if v.f ≠ -1 : return v.f
  f1 = g(v)
  f2 = v.fon
  for u in v.emp:
    f2 += g(u)
  v.f = max( f1, f2)
  return v.f
```

Problem plecakowy

Dane: $I = \{0, \dots, n-1\}$ - przedmioty

$w: I \to \mathbb{N}$ - wagi

$p: I \to \mathbb{N}$ - ceny/profity

$B \in \mathbb{N}$ - maks. waga

Zadanie: Znaleźć podzbiór I o maksymalnej sumarycznej cenie i łącznej wadze nie przekraczającej B

① Funkcja do obliczania

$f(i, b)$ = maksymalna suma cen przedmiotów ze zbioru $\{0, \dots, i\}$ nie przekraczających łącznej wagi b

wynik: $f(n-1, B)$

② Sformułowanie rekurencyjne

$$f(i,b) = \max\Big(\underbrace{f(i-1, b)}_{\text{nie bierzemy } i\text{-go przedmiotu}}, \underbrace{f(i-1, b-w(i)) + p(i)}_{\text{bierzemy } i\text{-ty przedmiot}} \Big)$$

$b-w(i)$ o ile ≥ 0

$$f(0,b) = \begin{cases} p(0), & w(0) \leq b \\ 0, & w(0) > b \end{cases}$$

③ Implementacja

```
def knapsack(W, P, B):
    n = len(W)
    F = [[0 for b in range(B+1)] for i in range(n)]
    for b in range(W[0], B+1):
        F[0][b] = P[0]
    for b in range(B+1):
        for i in range(1, n):
            F[i][b] = F[i-1][b]
            if b - W[i] >= 0:
                F[i][b] = max(F[i][b],
                              F[i-1][b-W[i]] + P[i])
    return F[n-1][B]
```