SW-zadanie 3

Denis Firat

March 2021

1 Zadanie 1

1.1 Program WITTI

```
import numpy
import re
import timeit
start = timeit.default_timer()
p = []
w = []
d = []
with open("dataPY.txt","r") as data:
    for n in range(16):
        s=(data.readline()).split()
        p.append(int(s[0]))
        w.append(int(s[1]))
        d.append(int(s[2]))

n=len(p)
N=1<<n
F=[0]
ListaL=[[]]

for set in range(1,N):
    sumaP=0
    b=1
    c=0
    #####zliczanie sumy####
    for j in range(n):
        if (set&b):
            c=c+p[j]
        b*=2
    F.append(99999999)
    ListaL.append([])
```

```
    b=1
    for j in range(n):
        if (set&b):
            if (F[set]>F[set - b] + w[j] * max(c - d[j], 0)):
                pomocniczy=ListaL[set-b].copy()
                pomocniczy.append(j+1)
                ListaL[set]=pomocniczy.copy()
                F[set]=F[set - b] + w[j] * max(c - d[j], 0)
        b=b*2
print("Jedna z najkr tszych kombinacji: ",ListaL[-1])
print("O d ugo ci: ",F[-1])
stop = timeit.default_timer()
print('Time: ', stop - start)
```

1.2 Program wszystkie permutacje

```
import numpy
import re
import itertools
import timeit
start = timeit.default_timer()
p = []
w = []
d = []
with open("dataPY.txt","r") as data:
    for n in range(10):
        s=(data.readline()).split()
        p.append(int(s[0]))
        w.append(int(s[1]))
        d.append(int(s[2]))

wektorIndeksow=range(0,len(p))
listaPermutacji=list(itertools.permutations(wektorIndeksow))
sumaCzasuMin = 99999999
sumaKarMin = 99999999
permutacjaMin = ()
for permutacja in listaPermutacji:
    sumaKar=0
    sumaCzasu=0
    for proces in permutacja:
        sumaCzasu+=p[proces]
        sumaKar=sumaKar + max(0, (sumaCzasu-d[proces])*w[proces])

    if sumaKar<sumaKarMin:
        permutacjaMin = permutacja
        sumaCzasuMin = sumaCzasu
        sumaKarMin = sumaKar
        numerPermutacji = listaPermutacji.index(permutacja)
print(permutacjaMin)
print(sumaCzasuMin)
print(sumaKarMin)
stop = timeit.default_timer()
print('Time: ', stop - start)
```

1.3 Wprowadzenie do problemu

Problem polega na kolejkowaniu zadań wykonywanych na jednej maszynie, które maja swoje deadliny i kary za przekroczenie tych deadlinów. Jak to zwykle w przemyśle bywa, opóźnienia sie zdarzenia, a nie każda sztuke detalu da sie wykonać na czas. Kluczowe w takich momentach jest podejmowanie decyzji, które zadania wykonać przed innymi by zminimalizować poniesione kary.

1.4 Opis dynamicznego programowania

1.4.1 Podejście rekurencyjne/całościowe

Najbardziej intuicyjne(ale zdecydowanie nie najszybsze) jest podejście rekurencyjne, które w swoim zamyśle ma sprawdzić wszystkie możliwe permutacja procesów i znaleźć permutacje optymalna. Wszytko byłoby dobrze gdyby nie byłby to problem n!, wiec czas działania programu rośnie niewyobrażalnie szybko.

1.4.2 Algorytm zapamietujacy

Algorytm zapamietujacy zapisuje w trakcie działania programu optymalne permutacje podzbiorów ,a nastepnie używa ich do obliczania "ceny" kolejnych podzbiorów, aż do uzyskania najlepszej permutacji, której kara za opóźnienia jest najmniejsza.
Na przykład:
...
W trakcie działania programu doszliśmy do podzbioru K1,K2,K3. Jak wyznaczyć optymalna permutacje? Dajemy K1 na koniec i sprawdzamy ile wynosiła optymalna kara podzbioru K2,K3(na całe szczeście już to wcześniej wyliczyliśmy), sumujemy je razem(kare za K1 na końcu i optymalna permutacje K2,K3 i lecimy dalej. Dajemy K2 na koniec, nastepnie sumujemy kare za K2 na końcu i optymalna permutacje K1 i K3. Na koniec dajemy K3 na koniec, sumujemy kare za K3 na końcu i optymalna permutacje K2 i K1. Sprawdzamy, która z tych sum jest minimalna. Uzyskujemy optymalna permutacje dla K1, K2, K3, K4. Możemy jej teraz używać do obliczania zbiorów nadrzednych, aż dojdziemy do pełnego zbioru.
...
Dzieki wracaniu sie do już obliczonych optymalnych permutacji podzbiorów znaczaco obniżamy złożoność obliczeniowa i zamiast problemy n! mamy problem $2^n \cdot n$. Użyłem kalkulatora graficznego aby przedstawić różnice miedzy tymi dwoma złożonościami

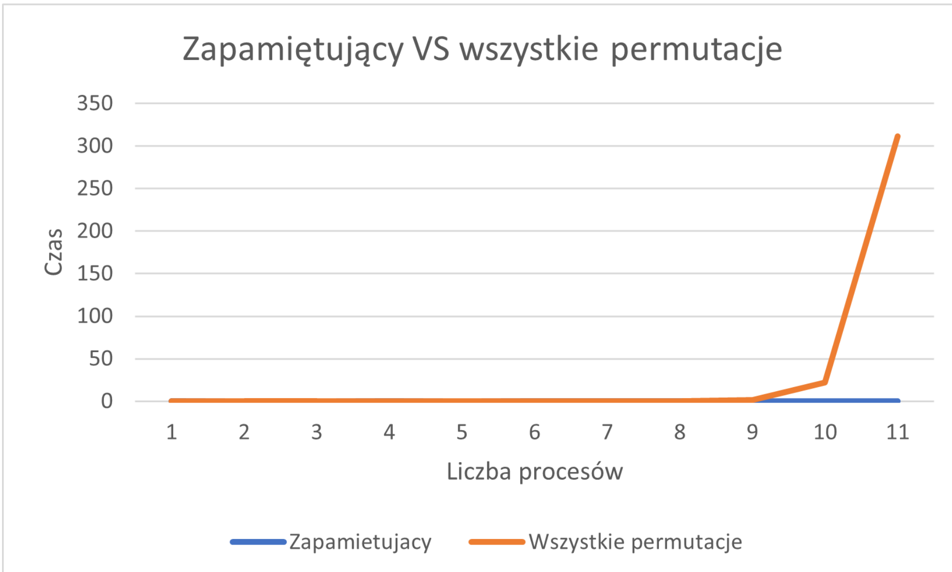

Figure 1: Porównanie działania programu

n	Zapamietujacy	Wszystkie permutacje
1	0.0016872	0.001632
2	0.0008686	0.001296
3	0.0009831	0.0011719
4	0.001352	0.0017348
5	0.001054	0.0014652
6	0.0013709	0.003433
7	0.0016617	0.0398061
8	0.0033106	0.2725817
9	0.0048883	1.7639431
10	0.0086738	21.8224394
11	0.0362856	311.4715227

Figure 2: Czas trwania programów w sekundach