Programowanie sieciowe– algorytmy CPM, PERT

Zadanie 1

Do realizacji wybrałem algorytm PERT, z którego pomocą wyznaczyłem termin realizacji oraz ścieżkę krytyczną przedsięwzięcia o minimum 20 łukach. Algorytm zrealizowałem łącząc metody programu Excel oraz własnoręcznie napisane funkcje w języku Python.

Kod z języka Python wykorzystany do wyznaczenia wartości: 'Pw', 'Pp', 'Kw', 'Kp', 'Zc', 'Zs', 'Zn'

```
import pandas as pd
import numpy as np
file_name = "Zeszyt2.xlsx"
df = pd.read_excel(file_name)
print(df)
```

```
# ustalenie Pw

for id, row in df.iterrows():
    if row['i'] == 1:
        df["Pw"][id] = 0
        df["Kw"][id] = df["Pw"][id] + df['t_o'][id]
    else:
        arrival_time = df['t_o'].where(df['j'] == row['i']) + df['Pw']
.where(df['j'] == row['i'])
        df["Pw"][id] = np.max(arrival_time)
        df["Kw"][id] = df["Pw"][id] + df['t_o'][id]

print(df)
```

```
# obliczenie Pp i Kp ostaniego elementu
last_activities = df["Kw"].where(df['j'] == np.max(df['j']))
for id, row in df.iterrows():
    if row['j'] == np.max(df['j']):
        df["Kp"][id] = np.max(last_activities)
        df["Pp"][id] = df["Kp"][id] - df['t_o'][id]


for id, row in df.iloc[::-1].iterrows():
    if not row['j'] == np.max(df['j']):
        arrival_time = df['Kp'].where(df['i'] == row['j']) - df['t_o']
.where(df['i'] == row['j'])
        df["Kp"][id] = np.min(arrival_time)
        df["Pp"][id] = df["Kp"][id] - df['t_o'][id]

print(df)
```

```python
# obliczenie Z
for id, row in df.iloc[::-1].iterrows():
    Tj_w = np.max(df['Kw'].where(df['j'] == row['j']))
    if row['i'] == 1:
        Ti_p = 0
    else:
        Ti_p = np.max(df['Kp'].where(df['j'] == row['i']))
    df["Zc"][id] = df["Kp"][id] - df["Kw"][id]
    df["Zs"][id] = Tj_w - df["Kw"][id]
    df["Zn"][id] = Tj_w - Ti_p - df["t_o"][id]


print(df)
```

```python
# ścieżka krytyczna
paths_lst = list()

critical_path = df.query('Zc == 0')
critical_path = critical_path[['i', 'j']]

critical_dic = dict()
for id, row in critical_path.iterrows():
    if row['i'] in critical_dic.keys():
        critical_dic[row['i']].append(row['j'])
    else:
        critical_dic[row['i']] = [row['j']]
print(critical_dic)

from typing import List, Dict
def dfs_iterative(G: Dict[int, List[int]], s: int, last) -> List[int]:
    S = list()
    S.append(s)

    visited = list()
    while S:
        v = S.pop()
        if v not in visited:
            visited.append(v)
            if v == last:
                break
            if v in G.keys():
                for u in G[v][::-1]:
                    S.append(u)
    return visited

ordered_cp = dfs_iterative(critical_dic, 1, np.max(df['j']))
print(ordered_cp)
```

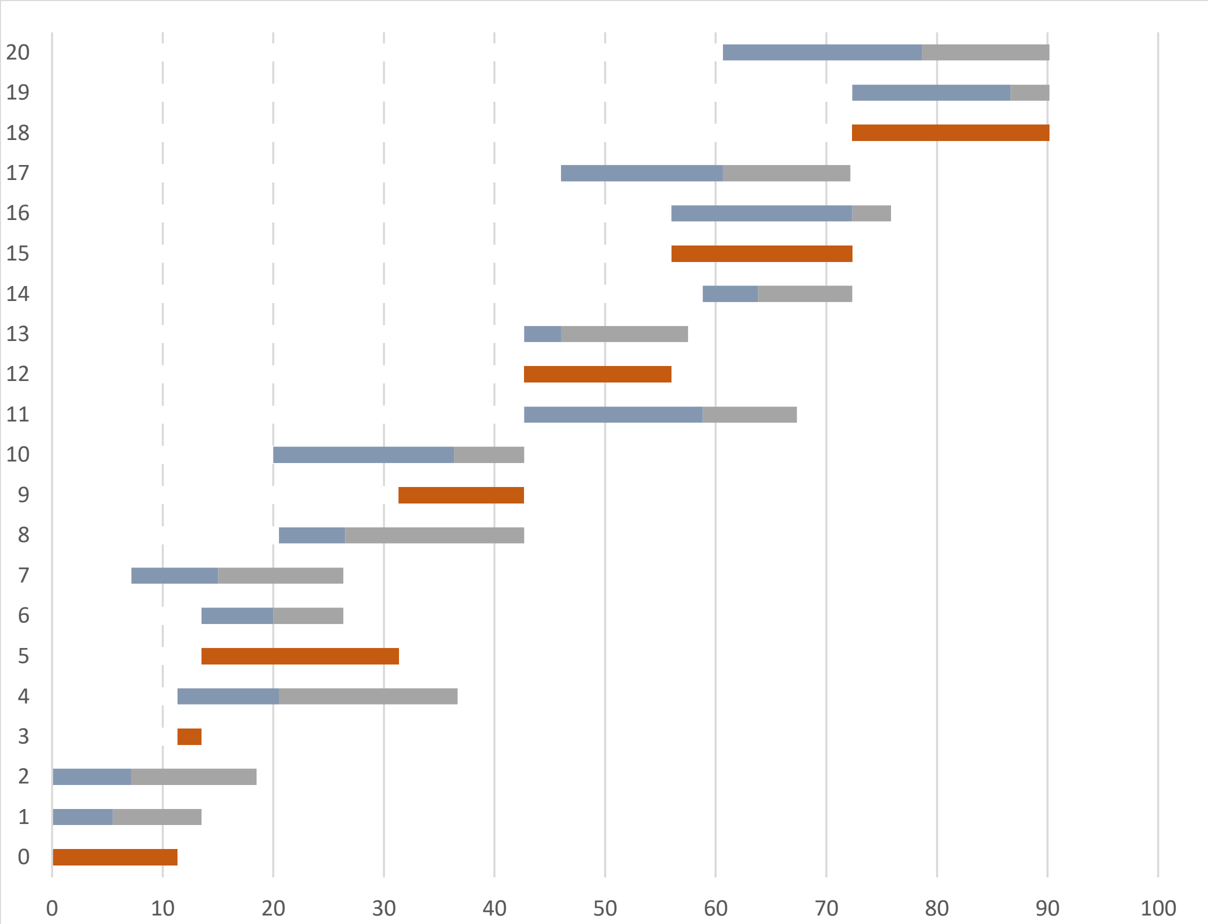
W programie Excel wyliczyłem za pomocą odpowiedniej formuły czas t_o oraz wariancje.

Zdefiniowane przedsięwzięcie:

i	j	t_c	t_m	t_p	t_o
1	2	13	14	15	11,33333
1	3	5	10	15	5,5
1	4	7	11	19	7,166667
2	3	2	2	2	2,166667
2	5	10	10	10	9,166667
3	6	20	21	22	17,83333
3	7	4	16	16	6,5
4	7	5	20	23	7,833333
5	8	5	8	11	6
6	8	12	12	12	11,33333
7	8	18	18	30	16,33333
8	9	17	20	20	16,16667
8	10	14	14	14	13,33333
8	11	1	5	15	3,333333
9	12	2	10	12	5
10	12	17	18	25	16,33333
10	13	15	15	15	16,33333
11	14	2	5	14	14,66667
12	15	18	20	28	17,83333
13	15	14	15	22	14,33333
14	15	18	21	24	18

Dla tak zdefiniowanego przedsięwzięcia czas wykonania ścieżki krytycznej wynosi 90,1(6), a wariancja całkowita jest równa 4,7(7), zatem posiłkując się tablicą dystrybuanty rozkładu normalnego, obliczam, że termin realizacji z prawdopodobieństwem 0,9 wynosi dokładnie 92,98, czyli zaokrąglając do pełnych wartości 93 dni.

Wykres Gantta omawianego przedsięwzięcia (Kolor czerwony – ścieżka krytyczna, kolor niebieski – pozostałe czynności, kolor szary– zapas całkowity)

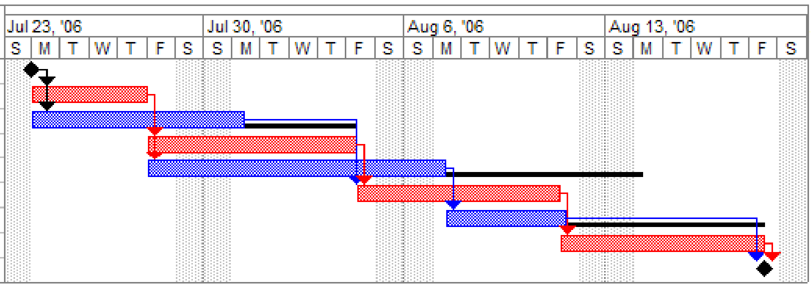


Ponieważ zadanie 2 opiera się o stopień automatyzacji zadania 1, pomijam rozważania na jego temat, gdyż istotne informacje zostały zawarte w opisie rozwiązania zadania 1, dodatkowo załączam wspomniany arkusz oraz plik z kodem pythonowskim.

Zadanie 3

Na wykresie Gantta rezerwę czasową można interpretować jako maksymalną odległość o jaką można przesunąć prostokąt reprezentujący dane zadanie bez powodowania opóźnienia w realizacji projektu. Często na wykresach nie jest zaznaczana rezerwa czasowa, jednak można spotkać się z wykresami zawierającymi taką daną co znacząco ułatwia jej odczytanie.

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



Rysunek 1 *diagram Gantta z zaznaczoną czarnym paskiem rezerwą czasową*