Sprawozdanie algorytm schrange

Kurs	Termin
Sterowanie procesami dyskretnymi	Czwartek 7:30-9:00
Skład grupy	**Kod grupy zajęciowej**
Dominik Koperkiewicz 254023, Marcel Konieczny 252966	Y00-39e
Prowadzący	**data**
Dr inż. Mariusz Makuchowski	12.05.2022

1 Cel zadania

Celem zadania było zapoznanie oraz zaimplementowanie algorytmu schrange aby następnie rozwiązać problem szeregowania zadań na maszynach.

2 Opis problemu

Mamy podane kilka zadań składających się z liczby R reprezentującej czas oczekiwania zanim zadanie będzie mogło być wykonane na maszynie, liczby P reprezentującej czas wykonywania na maszynie oraz liczby Q która reprezentuje jak długo należy poczekać po zakończeniu zadania. Rozwiązanie problemu polega na znalezieniu takiej kolejności wykonywania zadań, aby czas wykonywania wszystkich zadań od początku do końca był jak najkrótszy.

3 Kod programu

```
1  int schrange(int N, int* R, int* P, int* Q, int* X) {
2
3      int tmp;          //Zmienna pomocnicza do przechowywania numeru
           zadania
4      int C = 0;        //Czas wykonania
5      int tmpQ;         //Pomocnicza do zmiennej Q
6      int numzad = 0; //pomocnicza do ilo ci dzia a
7
8
9      //Schrange z podzialem
10     int m = 0;
11   /*  int time = 0;
12     int shortTime = 0;
```

```
13      bool petla = 1;
14      while (petla) {
15          shortTime = -1;
16          tmp = -1;
17          //numzad=0;
18          for (int i = 0;i < N;i++) {
19              if (R[i] <= time && P[i] > 0 && Q[i] > shortTime) {
20                  tmp = i;
21                  shortTime = Q[i];
22              }
23              / if (P[i] == 0) {
24                   numzad++;
25                  //cout<<numzad<<" ";
26                   if (numzad == N-1)
27                       return m;
28               }/
29          }
30          if (tmp != -1) {
31              P[tmp] = P[tmp] - 1;
32              time++;
33              // cout << tmp << " ";
34              if (P[tmp] == 0) {
35                  if (m < time + Q[tmp])
36                      m = time + Q[tmp];
37                  P[tmp] = 0;
38                  X[numzad]=tmp+1;
39                  numzad++;
40
41                  if (numzad == N)
42                      petla = 0;
43              }
44          }
45          else
46              time++;
47      }
48      return m;*/
49
50      //Schrange bez podzia u
```

```
51          int k = 0;
52          for (int i = 0;i < N;i++) {
53              tmpQ = 100000;
54              if (i == 0) {
55                  for (int j = 0;j < N;j++) {
56                      if (R[j] < tmpQ && P[j] != 0) {
57                          tmpQ = R[j];
58                          tmp = j;
59                      }
60                  }
61              }
62              else {
63                  k = m; //Aktualny czas
64                  tmpQ = -1;
65                  while (tmpQ < 0) {
66                      for (int j = 0;j < N;j++) {
67                          if (R[j] <= k && P[j] != 0) {
68                              if (tmpQ >= 0 && Q[tmp] > Q[j]) {
69                                  continue;
70                              }
71                              tmpQ = Q[j];
72                              tmp = j;
73                          }
74                      }
75                      k++;
76                  }
77              }
78              X[i]=tmp;
79              m = max(R[tmp], m) + P[tmp];
80              C = max(C, m + Q[tmp]);
81              P[tmp] = 0;
82          }
83          return C;
84  }
```

4 Opis działania programu

4.1 Schrage bez podziału

Na samym początku implementujemy pętle, która iteruje pomocniczą zmienną symbolizującą upływ czasu. Podczas wykonywania kolejnych iteracji pętli znajdziemy element który jest mniejszy lub równy wartości R jednego z pozostałych zadań, zadanie to wykonujemy. Jeżeli zdaży się sytuacja, że znalezione zostanie kilka zadań które mają R mniejsze niż zmienna symbolizująca czas, to wybieramy zadanie z największą wartością parametru Q. Czas wykonywania całego problemu jest obliczany na bieżąco.

4.2 Schrage z podziałem

W pętli z każdą koleją iteracją zwiększamy wartość czasu o 1. Iteracja ta odzwierciedla pomocniczy czas za pomocą którego wybieramy poszczególne zadania do wykonania. W pętli znajduje się kolejna pętla wybierająca zadanie (spośród tych które nie zostały jeszcze skończone) zawierające taki sam lub mniejszy czas R niż wartość pomocniczego czasu. Po wybraniu zadania zmniejszamy ilość potrzebnego czasu do wykonania tego zadania (P-1) i przystępujemy do kolejnej iteracji pętli głównej.

5 Wyniki

5.1 Wyniki dla schrange bez podziałem

```
1  Plik danych:  data.000
2  Laczny czas 283
3
4  Plik danych:  data.001
5  Laczny czas 3109
6
7  Plik danych:  data.002
8  Laczny czas 3708
9
10 Plik danych:  data.003
11 Laczny czas 3342
12
13 Plik danych:  data.004
14 Laczny czas 3235
15
```

```
16 Plik danych:  data.005
17 Laczny czas 3625
18
19 Plik danych:  data.006
20 Laczny czas 3446
21
22 Plik danych:  data.007
23 Laczny czas 3862
24
25 Plik danych:  data.008
26 Laczny czas 3645
27
28 Laczny czas: 28255
```

5.2 Wyniki dla schrange z podziałem

```
 1 Plik danych:  data.000
 2 Laczny czas 221
 3
 4 Plik danych:  data.001
 5 Laczny czas 3026
 6
 7 Plik danych:  data.002
 8 Laczny czas 3654
 9
10 Plik danych:  data.003
11 Laczny czas 3309
12
13 Plik danych:  data.004
14 Laczny czas 3172
15
16 Plik danych:  data.005
17 Laczny czas 3618
18
19 Plik danych:  data.006
20 Laczny czas 3439
21
22 Plik danych:  data.007
```

```
23 Laczny czas 3820
24
25 Plik danych:  data.008
26 Laczny czas 3633
27
28 Laczny czas: 27892
```

6 Wnioski

Pomimo że algorytm schrage nie odnajduje najbardziej optymalnego rozwiązania jest on w miarę prosty w implementacji. Porównując wyniki algorytmu schrage z algorytmem cariera(radzi sobie bardzo dobrze z problemem RPQ). Wyniki algorytmu schrage są 2% gorsze, wynik ten dla mniejszej ilości zadań nie będzie odczuwalny, lecz wraz z wzrostem zadań do wykonania, czas wykonywania będzie bardziej odczuwalny.

Proponowana ocena 3.5