Algoritmos e Estruturas de Dados

2020/2021

Trabalho 1

Generalized weighted job selection

David Ferreira, 98608 (50%);

Pedro Ferreira, 98620 (50%)

• Introdução

Este trabalho consiste em, dado um número de tarefas, cada uma com uma data de início, data de fim e lucro, e um dado número de programadores, encontrar o lucro máximo, sendo que os programadores só podem realizar uma tarefa de cada vez, e a sua duração depende da data de início e fim das mesmas.

• Descrição da solução computacional

Resolvemos este problema utilizando uma árvore binária, com dois estados para cada tarefa (feita ou não feita) e tentámos todas as combinações possíveis dado um número de tarefas e de programadores com uma pesquisa em profundidade. Por último, fomos comparando os resultados (lucro total) entre os ramos e devolvemos o ramo com maior lucro total.

Resultados (tabelas em apêndice)

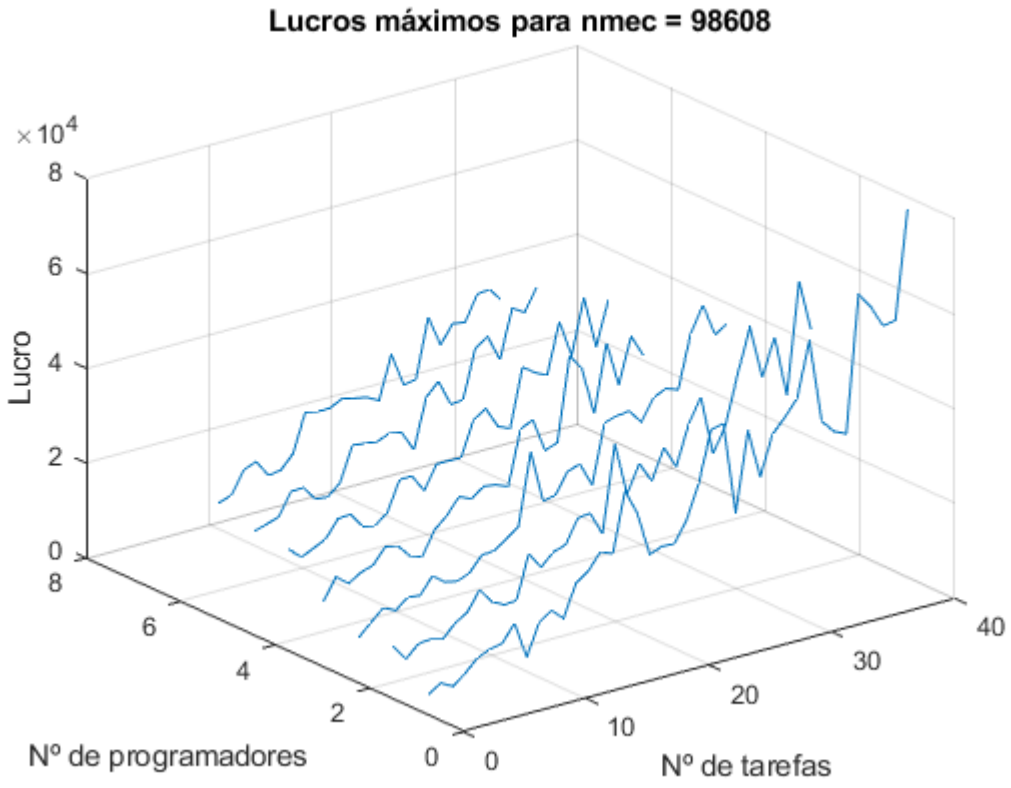

Gráfico 1

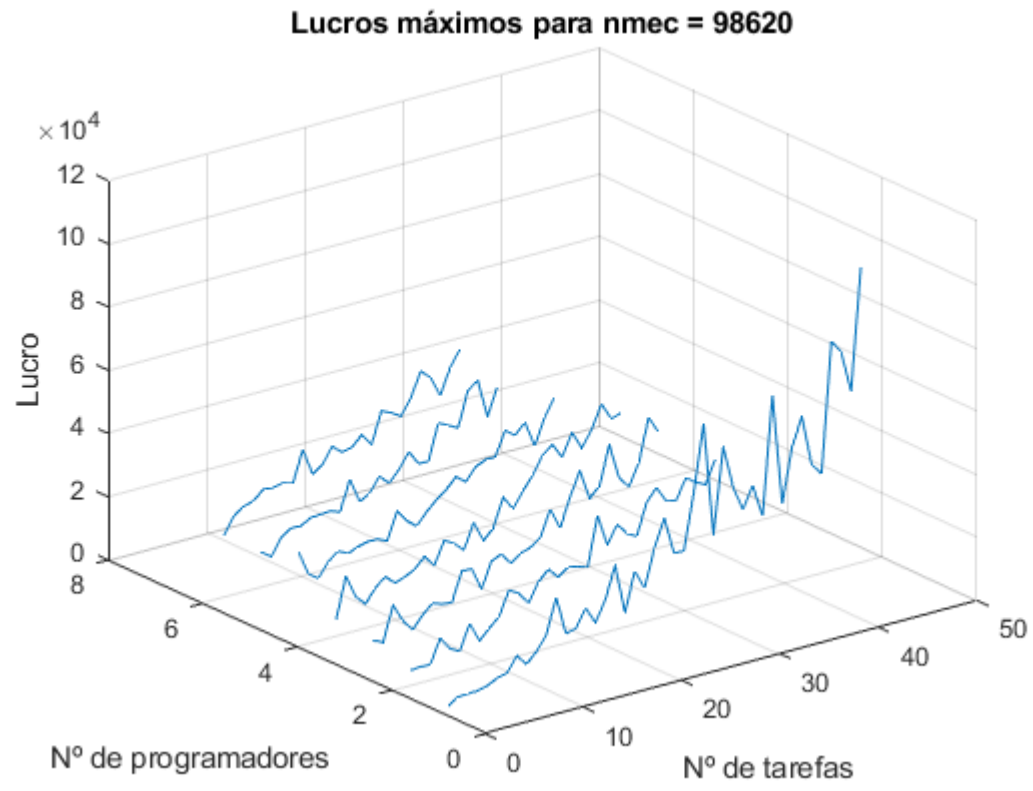

Gráfico 2

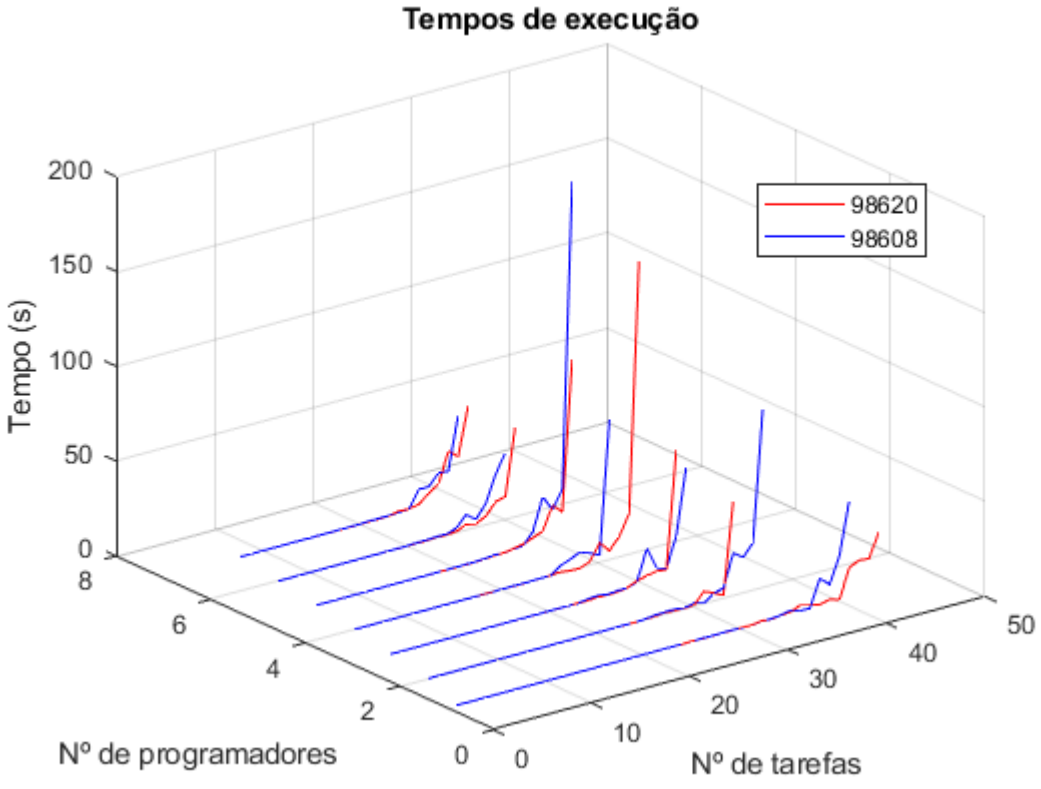

Gráfico 3

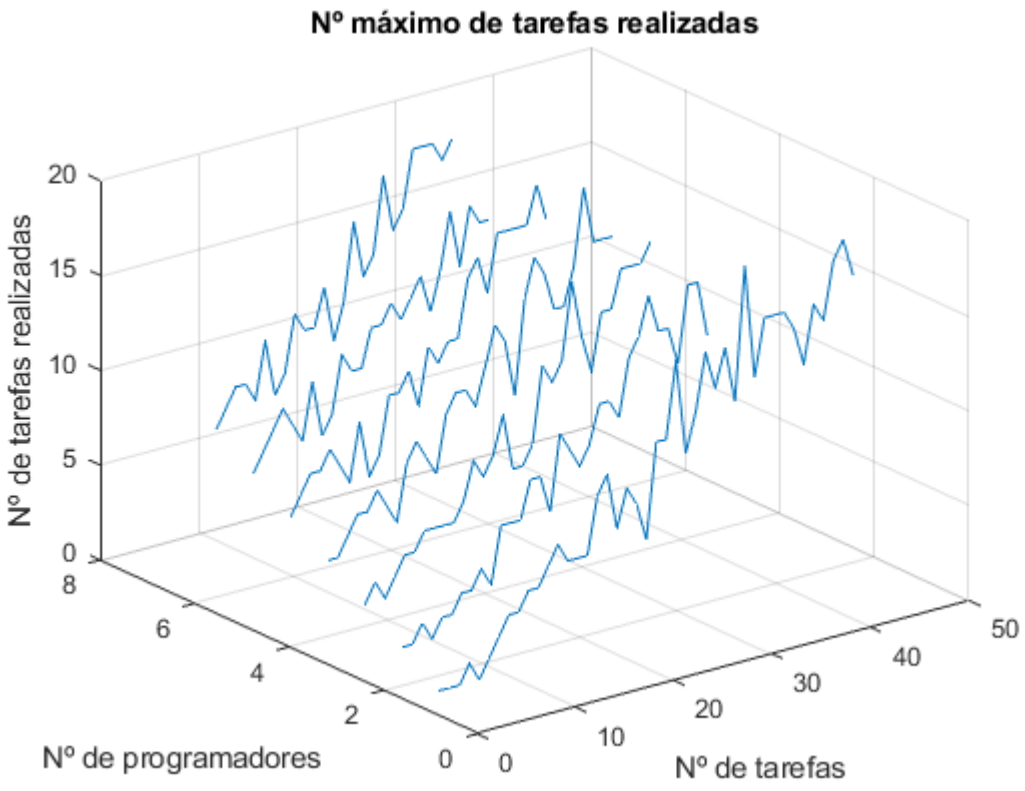

Gráfico 4

- **Conclusões**

Foram obtidos diferentes gráficos para os lucros máximos (gráficos 1 e 2) devido aos lucros serem gerados por números pseudo aleatórios, que dependem do número mecanográfico.

Também foi diferente o número de tarefas máximas para cada aluno, devido a capacidades do processador de computadores diferentes (computador com processador melhor demora menos tempo em cada iteração, logo poderá demorar mais iterações até alcançar o tempo máximo definido)

Relativamente ao gráfico 3, podemos observar que para cada número de programadores, os valores de tempo de execução definem aproximadamente uma função exponencial, visto que o número total de possibilidades é 2^n, sendo n o número de tarefas, logo enquanto n aumenta, o número total de possibilidades duplica, e consequentemente o tempo de execução também.

O gráfico 4 mostra os lucros máximos, com o lucro de todas as tarefas igual a 1, ou seja, está representado o número máximo de tarefas realizadas para cada caso.

Apêndice 1 – Código de resolução do problema (job_selection.c)

```
void assign_programmer_to_task(problem_t *problem, int task_number, int programmer_number) {
  problem -> task[task_number].assigned_to = programmer_number;
  problem -> busy[programmer_number] = problem -> task[task_number].ending_date;
}

int find_available_programmer(problem_t problem, int task_number) {
  for (int i = 0; i < problem.P; i++) {

    if (problem.busy[i] == -1 || problem.task[task_number].starting_date > problem.busy[i]) {
      return i;
    }
  }
  // no available programmer found
  return -1;
}

void solve_since_task(problem_t *problem, int task_number)
{
  problem_t problem_no_programmer_assigned = *problem;
  problem_t problem_programmer_assigned = *problem;

  // if there is no available task, return
  if (task_number >= problem -> T) {
    return;
  }

  // call function for next task
  solve_since_task(&problem_no_programmer_assigned, task_number + 1);

  // find available programmer
  int available_programmer = find_available_programmer(problem_programmer_assigned, task_number);
  // if there is an available programer:
  if(available_programmer != -1) {

    // assign programmer to task
    assign_programmer_to_task(&problem_programmer_assigned, task_number, available_programmer);

    // increment task profit to total_profit
    problem_programmer_assigned.total_profit += problem_programmer_assigned.task[task_number].profit;

    // call function for the next task
    solve_since_task(&problem_programmer_assigned, task_number + 1);
  }

  // assign problem with bigger profit
  if(problem_programmer_assigned.total_profit > problem_no_programmer_assigned.total_profit) {
    *problem = problem_programmer_assigned;
  }
  else {
    *problem = problem_no_programmer_assigned;
  }

}

// called in solve function
void solve_problem(problem_t *problem) {

  problem -> total_profit = 0;

  for(int i = 0; i < problem -> P; i++) {
    problem -> busy[i] = -1;
  }

  for(int i = 0; i < problem -> T; i++) {
    problem -> task[i].assigned_to = -1;
  }

  solve_since_task(problem, 0);
}
```

Apêndice 2 – Programa em python para automizar a recolha de dados e escrita em ficheiro excel

```python
import subprocess
import time
import xlwt

NMEC = 98620
MAX_TIME = 30
INF = 10000

book = xlwt.Workbook()
sheet1 = book.add_sheet(str(NMEC))
sheet1.write(0,0,'Tasks')
sheet1.write(0,1,'Programmers')
sheet1.write(0,2,'Profit')
sheet1.write(0,3,'Time')

num = 1
for P in range(1,8):
    for T in range(P, INF):
        start_time = time.time()

        execSc = ["./job_selection", str(NMEC), str(T), str(P), str(1)]
        proc = subprocess.Popen(execSc, stdout=subprocess.PIPE)
        # catch the output from terminal
        output, error = proc.communicate()

        end_time = time.time()

        output = output.decode('utf-8')
        profit = output[0]
        time = output[1]

        # write in table
        sheet1.write(num, 0, T)
        sheet1.write(num, 1, P)
        sheet1.write(num, 2, profit)
        sheet1.write(num, 3, time)

        num += 1

        time_total = end_time - start_time
        if time_total > MAX_TIME:
            break

book.save(str(NMEC))
```

Apêndice 3 – Programa de Matlab para criar os gráficos apresentados

```
plot3(Tasks,Programmers, Profit)
title("Lucros máximos para nmec = " + num2str(NMEC))
xlabel("N° de tarefas")
ylabel("N° de programadores")
zlabel("Lucro")
grid on

%%
% convert time values's format
temp = strcat(num2str(Time(:,1)),'.',num2str(Time(:,2)));

for i=1:length(temp)
dot = 0;
    for j=1:length(temp(i,:))
        if temp(i,j) == '.'
            dot = 1;
        end
        if dot == 1
            if temp(i,j) == ' '
                temp(i,j) = '0';
            end
        end
    end

end
Time = str2num(temp);
%%
a1 = plot3(Tasks,Programmers, Time, "r")
L1 = NMEC;

hold on

a2 = plot3(Tasks1,Programmers1,Time1, "b")
L2 = NMEC1;

title("Tempos de execução")
xlabel("N° de tarefas")
ylabel("N° de programadores")
zlabel("Tempo (s)")
grid on
%%

a1 = plot3(Tasks,Programmers, Profit, "r")
L1 = "98620";

hold on

a2 = plot3(Tasks1,Programmers1,Profit1, "b")
L2 = "98608";

title("Lucros máximos")
xlabel("N° de tarefas realizadas")
ylabel("N° de programadores")
zlabel("Tempo (s)")
grid on

```

Apêndice 3 – Dados dos gráficos 1 e 3

98620			
Tasks	Programmers	Profit	Time
1	1	990	0,000002
2	1	3090	0,000008
3	1	3255	0,000008
4	1	3334	0,000014
5	1	4141	0,000018
6	1	5692	0,000022
7	1	6397	0,000026
8	1	11068	0,000033
9	1	7529	0,000036
10	1	10761	0,00005
11	1	15064	0,000053
12	1	26234	0,000063
13	1	13845	0,000167
14	1	14669	0,000127
15	1	20242	0,000105
16	1	14771	0,000276
17	1	21148	0,001017
18	1	31299	0,001665
19	1	15385	0,000251
20	1	27736	0,000994
21	1	21427	0,001469
22	1	33483	0,00035
23	1	41947	0,003428
24	1	29946	0,007668
25	1	29954	0,088913
26	1	47445	0,019434
27	1	68333	0,032886
28	1	32536	0,159205
29	1	59441	0,074939
30	1	45777	0,190155
31	1	37889	0,071244
32	1	44758	0,889727
33	1	34075	0,319203
34	1	71557	1,128864
35	1	36340	1,487466
36	1	53608	3,552697
37	1	62154	2,096951
38	1	46144	1,209249
39	1	42509	2,473523
40	1	83502	0,832058
41	1	79199	16,424388
42	1	66111	18,932276
43	1	104621	18,278322
44	1	100898	30,692582
2	2	4657	0,000005
3	2	4927	0,000007
4	2	4820	0,00001
5	2	12294	0,000012
6	2	8128	0,000019
7	2	6501	0,000041

8	2	14382	0,000029
9	2	8089	0,000042
10	2	11237	0,000061
11	2	14161	0,000084
12	2	21618	0,000162
13	2	19663	0,000528
14	2	15909	0,000228
15	2	21649	0,000927
16	2	24777	0,00107
17	2	21478	0,000565
18	2	23757	0,003303
19	2	23172	0,004138
20	2	21976	0,004432
21	2	37546	0,003455
22	2	27359	0,026161
23	2	33079	0,019882
24	2	29303	0,060542
25	2	27650	0,082372
26	2	37502	0,342929
27	2	41361	0,814429
28	2	36191	0,672931
29	2	35523	0,509176
30	2	41950	0,522827
31	2	39665	6,429573
32	2	38101	4,254707
33	2	45419	1,526344
34	2	44198	50,91566
3	3	6388	0,000006
4	3	5012	0,000011
5	3	15864	0,000013
6	3	10091	0,000038
7	3	6400	0,000029
8	3	10172	0,000041
9	3	13098	0,000067
10	3	12057	0,000188
11	3	11681	0,000121
12	3	20769	0,000402
13	3	20591	0,000309
14	3	13363	0,000704
15	3	21312	0,001024
16	3	22697	0,00151
17	3	19006	0,005942
18	3	21388	0,005337
19	3	22328	0,003275
20	3	24729	0,005061
21	3	32705	0,022527
22	3	25861	0,051782
23	3	34795	0,042065
24	3	42292	0,32967
25	3	32773	0,397205
26	3	35876	0,283583
27	3	47973	0,282806
28	3	36666	1,174662
29	3	33086	3,364698
30	3	39808	4,806997

31	3	53281	5,690232
32	3	47797	5,150891
33	3	36589	67,38776
4	4	5012	0,000012
5	4	18237	0,000016
6	4	10767	0,000022
7	4	7713	0,000031
8	4	11923	0,000049
9	4	14641	0,000086
10	4	11781	0,000113
11	4	12914	0,000152
12	4	14446	0,000316
13	4	17626	0,001186
14	4	13952	0,000743
15	4	21008	0,00123
16	4	19234	0,003312
17	4	16286	0,007848
18	4	24103	0,009117
19	4	17640	0,009365
20	4	20719	0,065614
21	4	29622	0,098709
22	4	25030	0,050544
23	4	29893	0,035539
24	4	34010	0,340965
25	4	39495	0,444109
26	4	42039	0,918547
27	4	37294	0,65144
28	4	44112	0,32619
29	4	38475	2,096961
30	4	43436	10,565314
31	4	50679	5,44885
32	4	45086	11,226594
33	4	46232	21,661005
34	4	49585	153,51405
5	5	18719	0,000014
6	5	11138	0,000021
7	5	8856	0,000031
8	5	13319	0,000052
9	5	15876	0,000189
10	5	14192	0,000401
11	5	15486	0,000491
12	5	16265	0,000463
13	5	16222	0,000642
14	5	14881	0,000943
15	5	23408	0,00264
16	5	19507	0,004297
17	5	16989	0,007301
18	5	20611	0,00567
19	5	23631	0,030299
20	5	25998	0,022896
21	5	29254	0,047756
22	5	26770	0,056478
23	5	30691	0,260674
24	5	32131	0,717813
25	5	32154	0,844006

26	5	39633	1,597979
27	5	37204	2,644184
28	5	40554	4,598455
29	5	32260	6,581616
30	5	40295	18,833731
31	5	45849	14,059038
32	5	47290	93,120929
6	6	11138	0,000021
7	6	8856	0,000033
8	6	14134	0,000107
9	6	16187	0,000088
10	6	16049	0,000161
11	6	17814	0,000278
12	6	18063	0,001111
13	6	18289	0,000895
14	6	17114	0,00143
15	6	26566	0,003727
16	6	18668	0,005779
17	6	20480	0,00905
18	6	25066	0,018766
19	6	21983	0,027687
20	6	25493	0,059167
21	6	30214	0,07718
22	6	25579	0,166887
23	6	25632	0,458531
24	6	36863	0,240529
25	6	35236	0,634869
26	6	33530	2,872932
27	6	44462	1,920452
28	6	46940	4,454726
29	6	34562	10,663321
30	6	43402	12,29109
31	6	44395	47,338485
7	7	8856	0,000033
8	7	14134	0,000051
9	7	16187	0,000089
10	7	17162	0,000163
11	7	19971	0,00031
12	7	19605	0,000589
13	7	20246	0,001076
14	7	19277	0,001872
15	7	29100	0,004343
16	7	20540	0,007501
17	7	22683	0,013003
18	7	27594	0,026818
19	7	24911	0,029381
20	7	25608	0,058776
21	7	28747	0,202579
22	7	24797	0,171829
23	7	34570	0,322003
24	7	33172	1,411945
25	7	31228	1,072819
26	7	36267	2,184291
27	7	43773	6,517753
28	7	40972	10,04478

29	7	34368	25,874801
30	7	42486	21,609016
31	7	47361	46,878146

Apêndice 4 – Dados dos gráficos 2 e 3

98608			
Tasks	Programmers	Profit	Time
1	1	2614	0,000008
2	1	4303	0,000005
3	1	2905	0,000009
4	1	4989	0,000014
5	1	7512	0,000021
6	1	8646	0,000019
7	1	9144	0,000026
8	1	12671	0,000035
9	1	4810	0,000044
10	1	11440	0,000059
11	1	13253	0,0001
12	1	10920	0,000131
13	1	17627	0,000135
14	1	19463	0,000363
15	1	22851	0,00029
16	1	21720	0,001739
17	1	34258	0,001262
18	1	28797	0,000685
19	1	19477	0,004156
20	1	20433	0,002733
21	1	20303	0,011259
22	1	24552	0,007005
23	1	31715	0,014314
24	1	42238	0,00517
25	1	42869	0,03746
26	1	23301	0,181834
27	1	39923	0,072049
28	1	29485	0,179222
29	1	37774	0,268702
30	1	40611	0,128765
31	1	43953	0,24473
32	1	55430	0,579064
33	1	37669	0,541615
34	1	34863	1,240948
35	1	33660	2,237525
36	1	62329	0,504762
37	1	58958	0,703457
38	1	54250	14,838894
39	1	54803	10,406398
40	1	77704	24,027274
41	1	50462	51,643675

2	2	7329	0,000008
3	2	4030	0,000012
4	2	6537	0,000018
5	2	6847	0,000024
6	2	6144	0,000032
7	2	8806	0,000046
8	2	10440	0,000064
9	2	14447	0,000149
10	2	11234	0,00012
11	2	9937	0,000187
12	2	10120	0,000307
13	2	19231	0,000547
14	2	15689	0,000829
15	2	18196	0,002326
16	2	19035	0,001775
17	2	23965	0,002897
18	2	24237	0,011328
19	2	19198	0,026114
20	2	37414	0,008287
21	2	26206	0,02586
22	2	31936	0,049881
23	2	27503	0,048766
24	2	33862	0,15753
25	2	29123	0,255339
26	2	37388	0,924258
27	2	42443	1,113108
28	2	29785	1,272079
29	2	34961	0,624023
30	2	45162	1,621153
31	2	54371	0,266146
32	2	42978	4,07933
33	2	50846	5,429949
34	2	37875	22,371232
35	2	61238	18,660796
36	2	50024	25,47879
37	2	39746	94,37138
3	3	4030	0,000008
4	3	6537	0,000016
5	3	8897	0,000024
6	3	7669	0,000039
7	3	9702	0,000038
8	3	9413	0,0001
9	3	12829	0,000143
10	3	10835	0,000164
11	3	10269	0,000365
12	3	11280	0,000785
13	3	14337	0,001282
14	3	14543	0,002268
15	3	16337	0,000657
16	3	18355	0,004589
17	3	33377	0,004186
18	3	22240	0,008145
19	3	22788	0,022543

20	3	27097	0,087055
21	3	27927	0,096762
22	3	22801	0,079971
23	3	35227	0,02626
24	3	35994	0,801114
25	3	36313	1,191417
26	3	33282	0,473693
27	3	37770	1,20251
28	3	38941	1,89636
29	3	37999	2,817386
30	3	49019	19,52767
31	3	54415	7,370442
32	3	47480	5,809827
33	3	49157	21,472301
34	3	52566	56,242465
4	4	6537	0,000016
5	4	10859	0,000023
6	4	8697	0,00004
7	4	10351	0,00006
8	4	11159	0,000117
9	4	14353	0,0002
10	4	13865	0,000318
11	4	11035	0,00062
12	4	10148	0,000795
13	4	15081	0,001596
14	4	17409	0,001484
15	4	20732	0,004153
16	4	19481	0,005275
17	4	21491	0,015613
18	4	21098	0,01963
19	4	20037	0,0407
20	4	31426	0,091224
21	4	32666	0,178643
22	4	25523	0,119046
23	4	26484	0,212224
24	4	44083	0,358806
25	4	40584	0,426435
26	4	30429	4,97805
27	4	44726	6,094557
28	4	35085	8,495383
29	4	44645	6,464625
30	4	39933	4,220937
31	4	47590	75,133856
5	5	12022	0,000022
6	5	9698	0,000037
7	5	10875	0,000062
8	5	12208	0,000109
9	5	15728	0,0002
10	5	15951	0,000482
11	5	12615	0,000702
12	5	12100	0,001295
13	5	14214	0,002324
14	5	20413	0,00287

15	5	20488	0,004116
16	5	16668	0,005462
17	5	21585	0,014705
18	5	21586	0,025618
19	5	21517	0,04109
20	5	28776	0,101356
21	5	30478	0,104225
22	5	26027	0,225452
23	5	24892	0,414978
24	5	37103	1,04735
25	5	35283	0,616214
26	5	33969	1,393707
27	5	44573	1,885615
28	5	36693	7,722123
29	5	48297	24,713906
30	5	37178	16,700023
31	5	46490	25,904641
32	5	46926	186,71749
6	6	10584	0,00004
7	6	11343	0,000065
8	6	12208	0,000083
9	6	16847	0,000189
10	6	16969	0,000345
11	6	13854	0,000652
12	6	13605	0,001274
13	6	15923	0,002741
14	6	23094	0,005148
15	6	22771	0,007256
16	6	22439	0,010196
17	6	23489	0,023496
18	6	22966	0,032568
19	6	18606	0,068416
20	6	28966	0,092017
21	6	31652	0,359544
22	6	26021	0,443337
23	6	26411	0,739293
24	6	36518	0,731402
25	6	38224	2,302079
26	6	32717	8,198823
27	6	42832	4,267415
28	6	41234	11,089856
29	6	45896	25,19616
30	6	36925	35,305575
7	7	11343	0,000067
8	7	12208	0,000116
9	7	16847	0,000174
10	7	17874	0,000368
11	7	14294	0,000629
12	7	14618	0,001573
13	7	17446	0,002823
14	7	25451	0,005748
15	7	24896	0,011183
16	7	24939	0,019593

17	7	26210	0,040435
18	7	25610	0,051147
19	7	25094	0,107033
20	7	23721	0,159425
21	7	32765	0,393415
22	7	25644	0,413995
23	7	26036	0,472748
24	7	38529	0,885273
25	7	31927	1,087033
26	7	35735	10,028407
27	7	35262	9,998951
28	7	40555	15,914303
29	7	40810	15,362334
30	7	37728	43,603314