UNIVERSIDADE FEDERAL DE SÃO CARLC
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Bacharelado em Ciência da Computação
Estudante: Gabriela Vanessa Pereira Alves Mattos. RA: 551570

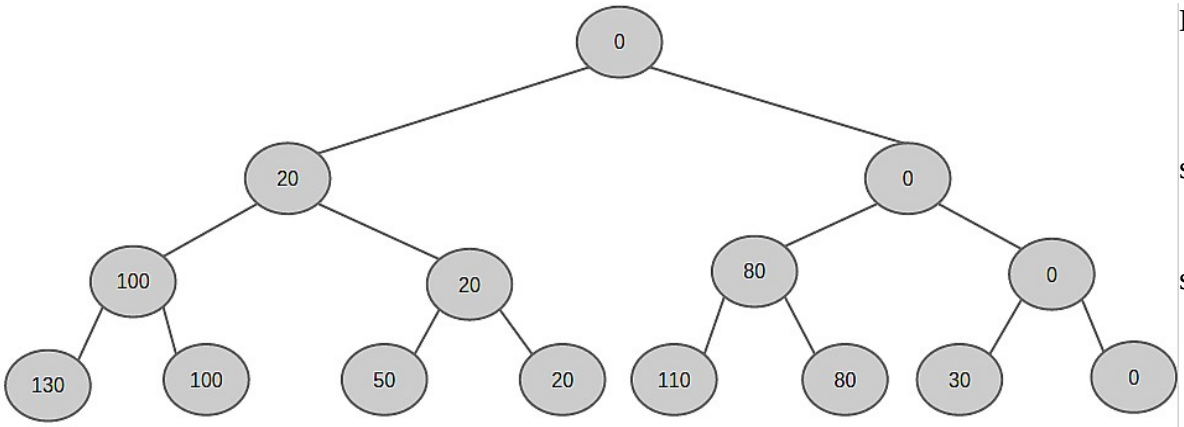
Problema B – Índice Carrasco Mamata

Implementado por força-bruta.

Para implementar esse problema foi utilizado uma árvore binária, onde em cada nível seleciona ou não uma turma informada pelo usuário, observe abaixo.

Entrada:	Saída:
5 seg 8 12 20 seg 8 12 80 ter 14 18 30 ter 16 18 60 qui 10 12 10	150

Usando o exemplo acima, será apresentada a árvore binária até a terceira linha digitada, ou seja, até ter 14 18 30.



Disciplina

seg 8 12 20

seg 8 12 80

ter 14 18 30

Informações importantes da árvore

- Cada nível terá a escolha ou não da turma;
- Nível determina turma escolhida;
- O somatório do indíce é o que está contido em cada nodo;

O algoritmo foi implementado seguindo o modelo da árvore binária, onde haverá sempre uma chamada da função que não escolherá a disciplina (subárvore do lado direito) e outra que selecionará (subárvore do lado esquerdo). Observe abaixo o código que permite isso:

```
1 else
2 {
3         indiceCM(tam, i+1, sol, vet, sum, disc); //chamada da funcao sem contar uma opcao
4         sol = vet[i];
5         horaI = sol.c - 8;
6         horaF = sol.f - 8;
7         dia_atual=defineDia(sol.s);
8
9         for (j=horaI; j<horaF; j++)
10                horario[j][dia_atual]++; //soma 1 a cada vez que o horario foi ocupado
11
12        indiceCM(tam, i+1, sol, vet, sum + vet[i].ind, disc+1); //chamada da funcao contando uma opcao
13
14        for (j=horaI; j<horaF; j++)
15                horario[j][dia_atual]--; //subtrai 1 para eliminar o horario que nao esta mais ocupado
16
17        sum -= vet[i].ind;      //subtrai o indice do disciplina que nao sera mais escolhida na proxima chamada
18 }
```

Na linha 3 foi realizada a chamada recursiva da função, onde essa chamada é para percorrer a subárvore direita, ou seja, a escolha de não colocar o *vet[i]* (vetor da estrutura que contém as informações das disciplinas), disciplina da posição i. Já na linha 12 chamasse a função escolhendo o *vet[i]*, ou seja percorrendo a subárvore esquerda, de acordo com a árvore apresentada acima.

Para entender como estão sendo definido os horários das turmas escolhidas, é necessário entender o funcionamento da matriz *horario*. Observe a imagem a baixo

	seg	ter	qua	qui	sex	sab	dom
08	0	0	0	0	0	0	0
09	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	2	0	0
17	0	0	0	0	2	0	0
18	1	0	0	0	2	0	0
19	1	0	0	0	0	0	0
20	1	0	0	0	0	0	0
21	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0

Trata-se de uma matriz de dimensão 15x7, onde as linhas representaram os horários (índice+8) e as colunas os dias da semana (com a função *defineDia* enumera os dias de 0 a 6). Logo as linhas 5, 6, 7 estão determinando os índices. Já as linhas 8 e 9 somam 1, isso acontece para mostrar que a turma foi selecionada, as linhas 14 e 15 subtrai 1 quando a turma já não está mais selecionada. Ou seja, quando 0 na posição da matriz representa que horário está livre, quando 'n' horário foi ocupado por 'n' turmas.

Observe também que nas linhas 12 e 17 está seguindo a mesma lógica de chamar a função e somar o índice da disciplina e depois em 17 subtrair, pois essa já não é mais escolhida.

Agora vamos entender como é definido o índice final que será apresentado, observando o trecho do código abaixo.

```
1 if(i>=tam)
2 {
3         if(horariodisponivel() && sumM <= sum)
4                 if (disc >= quantM)
5                 {
6                     quantM = disc; //salva o maior numero de disciplinas
7                     sumM = sum;    //salva o maior numero do somatorio dos indices que passam pelas condicoes
8                 }
9 }
```

Quando percorrer até o último nó de um galho da árvore faremos três testes para selecionar ou não o somatório dos índices do CM das turmas escolhidas.

Primeiramente verificamos o horário a partir da função *horariodisponivel()* que verifica se tem algum dos horários com valor maior que 1, caso exista é porque das turmas selecionadas houve conflito de horário, então não é um índice que pode ser escolhido.

Logo depois verificamos se esse índice é o maior do que o que já foi escolhido quando percorreu outro galho. Por último conferimos se essas turmas escolhidas para esse indíce são as que possuem o maior número de disciplinas selecionadas.

Resumindo, esse percurso da árvore só será selecionado quando não existir horário conflitantes, possuir um maior valor de índice acumulado e o maior número de disciplinas selecionadas.

Analisando o algoritmo e a árvore, percebe-se que na primeira turma escolhida a chamada

acontecerá duas vezes, na segunda quatro vezes, logo assim que for aumentando a quantidade de turmas aumentará o número de chamadas exponencialmente. Ou seja, cada nível da árvore (chamamos de k) terá 2^k chamadas da função *indiceCM()*. Portanto, analisando em relação ao tempo temos a ordem de crescimento igual a 2^k, sendo k igual ao *n* número de turmas informadas.

Em relação a memória nessa função, temos a ordem de crescimento constante também, não haverá necessidade de memória adicional. No *main* é utilizado memória adicional para criação da matriz de *horário*, de tamanho 16x7 e para criação de uma variável *sol* que guardará a turma que atual que está sendo inserida na árvore.

Com os testes realizados com as entradas informadas pelo professor foi possível checar a resposta das entradas B_0, B_1, B_2, B_3, B_4, B_107 e B_108, as demais demoraram muito tempo (mais de 3 horas) então não foi possível chegar no final. As que foram testadas deram os resultados certos (de acordo com os arquivos de saídas apresentados).