Estrutura de dados avançada

Mário pinto / 23506
(nrº , regime diurno)

Orientação de

Licenciatura em Engenharia em Sistemas Informáticos
Escola Superior de Tecnologia
Instituto Politécnico do Cávado e do Ave

Identificação do aluno

Mário pinto / 23506
Aluno número , regime diurno
Licenciatura em Engenharia em Sistemas Informáticos

Orientação

Resumo

Com este projeto de avaliação pretende-se sedimentar os conhecimentos introduzidos nas aulas da unidade curricular Estrutura de dados avançados.

Com este trabalho prático pretende-se sedimentar os conhecimentos relativos a definição e manipulação de estruturas de dados dinâmicas na linguagem de programação C. A essência deste trabalho reside no desenvolvimento de uma solução digital para o problema de escalonamento denominado Flexible Job Shop Problem (FJSSP). A solução a implementar deverá permitir gerar uma proposta de escalonamento para a produção de um produto envolvendo várias operações e a utilização de várias máquinas, minimizando o tempo as unidades de tempo necessário na sua produção.

Conteúdo

Lista de Figuras

1. Introdução

Nesta fase do trabalho é pedido para desenvolver uma solução digital para o problema de escalonamento denominado Flexible Job Shop Problem. Com este trabalho tencionamos aplicar tudo o aprendemos sobre estruturas e algoritmos.

1.1 Objetivos

A essência deste trabalho reside no desenvolvimento de uma solução digital para o problema de escalonamento denominado Flexible Job Shop Problem (FJSSP).

1. Definição de uma estrutura de dados dinâmica para representação de um conjunto finito de m jobs associando a cada job um determinando conjunto finito de operações;
2. Armazenamento/leitura de ficheiro de texto com representação de um process plan (considerar obrigatoriamente para efeito de teste o process plan da Tabela 1);
3. Inserção de um novo job;
4. Remoção de um job;
5. Inserção de uma nova operação num job;
6. Remoção de uma determinada operação de um job;
7. Edição das operações associadas a um job;
8. Cálculo de uma proposta de escalonamento para o problema FJSSP (obrigatoriamente limitado a um tempo máximo de processamento configurável), apresentando a distribuição das operações pelas várias máquinas, minimizando o makespan (unidades de tempo necessárias para a realização de todos os jobs). A proposta de escalonamento deverá ser exportada para um ficheiro de texto possibilitando uma interpretação intuitiva (utilizar por exemplo um formato tabular ou representação gráfica html, ou outra);
9. Representação de diferentes process plan (variando a quantidade de máquinas disponíveis, quantidade de job, e sequência de operações, etc) associando as respetivas propostas de escalonamento.

2. Funções e algoritmos

2.1 inserirJobs

inserir jobs na lista de jobs, e se não conter uma operação cria a operação

```
1  Job* inserirJobs(Job * jp, int id, int* operacao, int size){
2
3    Job *jb = (Job*) malloc(sizeof(Job));
4
5    if (jb!=NULL)
6    {
7      jb->id = id;
8      //copiar array
9      int sizearrayoperacao=sizeof(operacao);
10     for(int h=0;h<sizearrayoperacao;h++){
11       jb->operacao[h] = operacao[h] ;
12     }
13     jb->sizeOP = size;
14     jb->seguinte = jp;
15     return(jb);
16   }
17   else return(jp);
18 }
```

2.2 procuraOperacoesInt

Verifica se já existe essa mesma operação nesse job para não haver erro

```
1  int procuraOperacoesInt(Operation *op, int id){
2    if(op==NULL) return 0;
```

```
  else{
    Operation* aux = op;
    while(aux != NULL){
      if(aux->id == id){
        return 1;
      }
      aux= aux->seguinte;
    }
    return 0;
  }
}
```

2.3 removerJobs

remover jobs na lista de jobs, ao eliminar simplesmente remove o job da lista.

```
Job *removerJobs(Job *jp, Operation *op, int id){
  //se a lista ficar vazia
  //remove so o job
  if( jp == NULL) return NULL;

  if(jp->id == id){
    Job* aux = jp;
    jp=jp->seguinte;
    free(aux);
  }else{
    Job *aux=jp;
    Job *auxAnt = aux;
    while(aux && aux->id != id){
      auxAnt=aux;
      aux = aux->seguinte;
    }
    if(aux != NULL){
      auxAnt->seguinte= aux->seguinte;
      free(aux);
    }
  }

```

```
23   //retorna a lista job
24   return jp;
25 }
```

2.4 insOpJP

inserir operação em especifico job

```
1  //inserir operação em job
2  Job *insOpJp(Job * jp,int idOp,int idJp ){
3    Job *aux = procuraJob(jp, idJp);
4
5    //incrementa o tamanho
6    aux->sizeOP++;
7    aux->operacao[jp->sizeOP-1]=idOp;
8
9    return jp;
10 }
```

2.5 rmOpJp

remover operação em especifico job

```
1  Job *rmOpJp(Job * jp,int idOp,int idJp ){
2    Job *aux = procuraJob(jp, idJp);
3
4    for(int i=0;i<jp->sizeOP;i++){
5      if(idOp==aux->operacao[i]){
6
7        for(int f=i-1;f<jp->sizeOP-1;f++){
8          aux->operacao[i]=aux->operacao[i+1];
9        }
10     }
11   }
12   aux->sizeOP -=1;
13
```

```
14    return jp;
15  }
```

2.6 alteraOperacao

Alterar uma operação na lista de operações

```
1   Operation* alteraOperacao(Operation* op, int id,int* maq,int* temp,int
2     Operation* aux = procuraOperacoes(op, id);
3     if(aux != NULL){
4       aux->sizeMT=size;
5       // insere as maquinas de novo
6       // passa de um arrey para o outro /maquina/
7       for(int i=0; i<size;i++){
8       aux->maquina[i]=maq[i];
9       }
10
11      // passa de um arrey para o outro /maquina/
12      for(int j=0; j<size;j++){
13      aux->tempo[j]=temp[j];
14      }
15
16    }
17    return op;
18  }
```

3. Programa em C

3.1 Função Main

Inserir job com operações

Como eu insiro e peço ao user as informações.

```
case 1:
    system("clear");
    //inserir Job
    printf("####Inserir Job####\n");
    idCountJb=quantidadeJobs(jobs);
    idCountJb++;
    printf("Job nº%d\n",idCountJb);

    //condicao se nao houver operacoes nao deixar adicionar job
    if(quantidadeOperacoes(operacoes)==0){
printf("Não existe operações insira operações \npara poder inserir um j
        printf("Deseja adicionar operações? Y/N:");
        scanf("%s",cc);

        if(strcmp(cc,"Y")==0 || strcmp(cc,"y")==0){
            printf("###Inserir operações###\n");
            printf("Quantas Operações:");
            scanf("%d",&qtoperacoes);

            for(int f=0;f<qtoperacoes; f++){
                 //para ir buscar a quantidade de op
                idCountOp=quantidadeOperacoes(operacoes);
                idCountOp++;
                printf("Quantas maquinas para a operação nº %d:",f+1);
```

```
25                  scanf("%d",&qt);
26                  int maq[qt];
27                  int temp[qt];
28                      for(int i=0; i<qt; i++){
29                          printf("\nMaquina numero:");
30                          scanf("%d",&maq[i]);
31                          printf("tempo da maquina %d:",i+1);
32                          scanf("%d",&temp[i]);
33      operacoes=inserirOperacoes(operacoes,idCountOp,maq,temp,qt);
34                      }
35              }
36          }else{
37              break;
38          }
39      }
40
41      printf("Quantas operações deseja?\n");
42      scanf("%d",&qtOpCiclo);
43      for(int i=0; i<qtOpCiclo; i++){
44          listarOperations(operacoes);
45          printf("Qual operação deseja:");
46          scanf("%d",&aa[i]);
47          verificacao=procuraOperacoesInt(operacoes, aa[i]);
48          while (verificacao==0)
49          {
50              printf("Nao existe essa operacao");
51              printf("Qual operação deseja:");
52              scanf("%d",&aa[i]);
53              verificacao=procuraOperacoesInt(operacoes, aa[i]);
54          }
55
56      }
57      //inserir com tudo
58      jobs=inserirJobs(jobs,idCountJb,aa,qtOpCiclo);
59      //operacoes= inserirOperacoes(operacoes,1,bb,cc,3);
60      //operacoes= inserirOperacoes(operacoes,2,rr,tt,3);
61      //operacoes= inserirOperacoes(operacoes,3,kk,ii,1);
62      printf("Job predefenido inserido com sucesso!\n");
63      break;
```

Remover uma operação

Compara o id na lista e remove.

```
case 4:{
        int idRemover=0;
        system("clear");
        printf("###### remover operação #####\n\n");
        printf("Qual id da operação:");
        scanf("%d",&idRemover);
        removerOperacoes(operacoes,idRemover);
        break;
            }
```

Inserir uma operação num job

adiciona no job uma posição com o id da operação.

```
 case 11:{
    //ve quantas operacoes tem inseridas para ter o id correto

    //inserir operacoes e ver a quantidade de maquinas
    system("clear");
    //se if no case 1 for true vem para aqui!!!
    printf("###Inserir operações###\n");

    //para ir buscar a quantidade de op
    idCountOp=quantidadeOperacoes(operacoes);
    idCountOp++;
    printf("Quantas maquinas:");
    scanf("%d",&qt);


    for(int i=0; i<qt; i++){
        printf("\nMaquina numero:");
        scanf("%d",&maq[i]);
```

```
19          printf("tempo da maquina %d:",i+1);
20          scanf("%d",&temp[i]);
21      }
22
23      operacoes=inserirOperacoes(operacoes,idCountOp,maq,temp,qt);
24
25      break;
26  }
```

remover um job

Remove o job da lista dos jobs

```
1  case 12:
2   //TODO: acabar remover
3      //BUG: nao executa a parte de remover as operações !!!  direito
4      //adicionar changes
5          printf("Qual job deseja remover:");
6          scanf("%d",&idremover);
7          jobs=removerJobs(jobs,operacoes,idremover);
8  break;
```

remover uma operação de job

Remove o id do array de operações do job

```
1  case 14:
2      system("clear");
3      printf("### Remover OP em JP  ###\n");
4      listarJobs(jobs);
5      printf("Em qual job quer remover:");
6      scanf("%d",&idjobRm);
7      printf("Qual operacão quer remover:");
8      scanf("%d",&idOpRm);
9
10     jobs=rmOpJp(jobs,idOpRm,idjobRm);
11 break;
```

4. 4

4.1 Git Doxygen

GitHub: https://github.com/MarioJoao31/ProjetoIndividual-EDA
Doxygen vai em anexo, com a pasta html e com o ficheiro doxygen.

4.2 Conclusão

Com este trabalho consegui aplicar o que aprendi nas aulas sobre os algoritmos e sobre as estruturas.
Infelizmente não consegui realizar o trabalho na sua totalidade fazendo apenas 6 dos 9 objetivos Para alem deste trabalho ser desafiador foi bastante produtivo porque me pos a prova durante varios bugs durante a sua produção.

4.3 Bibliografia

https://www.geeksforgeeks.org/maximum-and-minimum-in-an-array/
https://stackhowto.com/c-program-to-search-an-element-in-an-array/
https://www.geeksforgeeks.org/find-the-minimum-distance-between-two-numbers/