DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Estruturas de Dados

Avaliação Comparativa de Algoritmos de Ordenação

Professores: Wagner Meira Jr
Eder Figueiredo

Avaliação de Algoritmos de Ordenação

- Métricas de Avaliação
  - Comparações
  - Movimentações
  - Chamadas de função
- Estratégia
  - Avaliação baseada em contadores
  - Instrumentação explícita
    - Avaliação não considera tempo
  - Plano de experimentos
    - Algoritmo
    - Tamanho do vetor a ser ordenado
    - Semente aleatória

Algoritmos

- Seleção
  - Iterativo
  - Recursivo
- Inserção
- QuickSort
  - Padrão
  - Mediana de 3
  - Inserção para partições pequenas (p.ex., 50)
  - Mediana de 3 e Inserção
- Shellsort
  - Utilizando sequência $n/2^i$

Instrumentação

```
typedef struct sortperf{
  int cmp;
  int move;
  int calls;
} sortperf_t;

void resetcounter(sortperf_t * s){
// Descricao: inicializa estrutura
// Entrada:
// Saida: s
  s->cmp = 0;
  s->move = 0;
  s->calls = 0;
}
```

Instrumentação

```
void inccmp(sortperf_t * s, int num){
// Descricao: incrementa o numero de comparacoes em
num
// Entrada: s, num
// Saida: s
  s->cmp += num;
}

void incmove(sortperf_t * s, int num){
// Descricao: incrementa o numero de movimentacoes
de dados em num
// Entrada: s, num
// Saida: s
  s->move += num;
}
```

Instrumentação

```
void inccalls(sortperf_t * s, int num){
// Descricao: incrementa o numero de chamadas de
função em num
// Entrada: s, num
// Saida: s
  s->calls += num;
}
char * printsortperf(sortperf_t * s, char * str,
char * pref){
// Descricao: gera string com valores de sortperf
// Entrada: s, pref
// Saida: str
 sprintf(str,"%s cmp %d move %d calls %d", pref,
s->cmp, s->move, s->calls);
 return str;
}
```

Seleção Recursivo

```
void initVector(int * vet, int size){
// Descricao: inicializa vet com valores aleatorios
// Entrada: vet
// Saida: vet
  int i;
  for (i=0; i<size; i++){
    vet[i] = (int)(drand48()*size);
  }
}
void swap(int *xp, int *yp, sortperf_t *s){
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
   incmove(s,3);
}
```

Seleção Recursivo

```
void recursiveSelectionSort(int arr[], int l, int r,
sortperf_t * s){
    int min = l;
    inccalls(s,1);
    for (int j = l + 1; j <= r; j++){
        inccmp(s,1);
        if (arr[j] < arr[min]) {min = j;}
    }
    if (min!=l) swap(&arr[min], &arr[l], s);
    if (l + 1 < r) {
        recursiveSelectionSort(arr, l + 1, r, s);
    }
}
```

Função main

```
parse_args(argc,argv,&opt);

// malloc with opt.size+1 for heapsort
vet = (int *) malloc((opt.size+1)*sizeof(int));

// initialize
resetcounter(&s);
srand48(opt.seed);
initVector(vet, opt.size);
vet[opt.size] = vet[0]; // for heapsort
```

Função main

```
retp = clock_gettime(CLOCK_MONOTONIC, &inittp);
switch (opt.alg){
   case ALGRECSEL:
      recursiveSelectionSort(vet,0,opt.size-1, &s);
      break;
}
retp = clock_gettime(CLOCK_MONOTONIC, &endtp);
clkDiff(inittp, endtp, &restp);
sprintf(pref,"alg %s seed %d size %d time %ld.%.9ld",
         num2name(opt.alg),
            opt.seed,opt.size,
            restp.tv_sec,restp.tv_nsec);
printsortperf(&s,buf,pref);
printf("%s\n",buf);
```

Submissão

- A submissão será feita por **VPL**. Certifique-se de seguir as instruções do tutorial disponibilizado no moodle.
- O seu arquivo executável **DEVE** se chamar **pa1.out** e deve estar localizado na pasta **bin**.
- Seu código será compilado com o comando:

  make all
- Você **DEVE** utilizar a estrutura de projeto abaixo junto ao Makefile :

  |- src
  |- bin
  |- obj
  |- include
  Makefile