Ordenação indireta com ponteiros

Algoritmos e Estruturas de Dados II

Ordenação indireta

- Quando os elementos a serem ordenados são grandes é melhor usar ordenação indireta
- Ordenar um vetor de ponteiros para os elementos

```
struct mem_page {
    char data[4096]; /* 1KB */
    int last_access;
};
struct mem_page v[65535]; /* 256MB */
```

Opção 1 – Ordenação direta

```
void insercao(struct mem_page *v, int n) {
  int i, j;
  struct mem_page aux;
  for(i = 1; i < n; i++)
    for(j = i; j >= 0; k--)
      if(v[j-1].last_access > v[j].last_access)
         troca(v[j-1], v[j]);
}

int main(int argc, char **argv) {
   ...
   insercao(v, 65535);
   ...
}
```

Opção 2 – Ordenação indireta

```
void insercao(void **v, int n,
              int(*compara)(void *a, void *b)) {
  int i, j; struct mem_page aux;
  for(i = 1; i < n; i++)
    for(j = i; j >= 0; k--)
      if(compara(v[j-1], v[i]) > 0)
        troca(v[j-1], v[j]);
}
int main(int argc, char **argv) {
   ...
   struct mem_page **ptr;
   ptr = malloc(65535, sizeof(struct mem_page *));
   for(i = 0; i < 65535; i++) ptr[i] = &v[i];
   insercao_ptr(ptr, 65535, compara);
   ...
}
```

Opção 2 – Ordenação indireta

```
int compara(void *a, void *b)
{
   struct mem_page *p1 = a;
   struct mem_page *p2 = b;
   if(a->last_access < b->last_access) return -1;
   if(a->last_access > b->last_access) return +1;
   return 0;
}
```