Comparativo entre os TADs

Prof. Bruno Travençolo

Comparativo

- O objetivo do comparativo é evidenciarmos as diferentes dos TADs aprendidos no curso com relação aos seguintes critérios
  - Tempo computacional
  - Espaço em memória
  - Aplicação
- Ver aula: 02.AnalisedeAlgoritmos_parte2.pdf

Comparativo

- Serão Comparados
  - Lista Sequencial Estática
  - Lista encadeada dinâmica com nó descritor
  - Lista duplamente encadeada
  - Fila
  - Pilha

Notação O (big-oh)

- A notação big-oh (O) descreve o limite assintótico superior de um algoritmo
- Mostra o limite do pior caso de um algoritmo

Funções importantes (1/3)

- Constante: $\approx$ O(1)
  - independente do tamanho de *n*, operações executadas um número fixo de vezes.
- Logarítmica: $\approx O(\log_b n)$
  - típica de algoritmos que resolvem um problema transformando-o em problemas menores.
  - para dobrar $\log_2 n$ é preciso fazer $\log_2 n^2$.
  - a base também muda pouco os valores: $\log_2 n \approx 20$ e $\log_{10} n \approx 6$ para $n = 1000000$.
- Linear: $\approx O(n)$
  - em geral, uma certa quantidade de operações é realizada sobre cada um dos elementos de entrada.
  - melhor situação para quando é preciso processar n elementos de entrada e obter n elementos de saída.

Lista Sequencial Estática

Operação	Complexidade	Sobre

push_front	O(N)	Inserir no início envolve o deslocamento de todos os elementos
		`for(i=li->qtd-1; i>=0; i--)`
		`  li->dados[i+1] = li->dados[i];`

push_back	O(1)	Temos um índice com acesso direto ao último elemento da lista `li->dados[li->qtd] = al;`
ordenada	O(N)	No pior caso envolve inserir no Início

pop_front	O(N)	Remover do início envolve mover todos os elementos
		`for(k=0; k< li->qtd-1; k++)`
		`  li->dados[k] = li->dados[k+1];`

pop_back()	O(1)	Basta modificar a variável de quantidade `li->qtd--;`
busca	O(N)	No pior caso o elemento está no final
Acesso (pos)	O(1)	Por estar em um vetor temos acesso em tempo constante

Lista Sequencial Estática

- Vantagens
  - Acesso rápido e direto aos elementos
  - Tempo constante pra acessar um elemento
  - Facilidade para modificar as informações
- Desvantagem
  - Definição prévia do tamanho do array
  - Dificuldade na inserção e remoção pois envolve deslocamento
- Uso
  - Listas pequenas
  - Inserção e remoção apenas no final da lista
  - Tamanho máximo da lista bem definido
  - A busca é a operação mais frequente

Lista Dinâmica Encadeada

Operação	Complexidade	Sobre
push_front	O(1)	Envolve apenas manipulação de alguns ponteiros
push_back	O(N)	Precisa percorrer toda a lista para chegar até ao último elemento. Pode ser reduzido para O(1) caso exista um ponteiro para o último elemento da lista (end)
ordenada	O(N)	No pior caso envolve inserir no final
pop_front	O(1)	Envolve apenas manipulação de ponteiros da cabeça da lista
pop_back()	O(N)	Precisa percorrer toda a lista para chegar até ao último elemento. Pode ser reduzido para O(1) caso exista um ponteiro para o último elemento da lista (end)
busca	O(N)	No pior caso o elemento está no final
Acesso (pos)	O(N)	No pior caso o elemento está no final

Lista Dinâmica Encadeada

- Vantagens
  - Melhor utilização da memória
  - Não é preciso definir previamente o tamanho da lista
  - Inserção e remoção sem necessidade de movimentar elementos
- Desvantagens
  - Acesso indireto aos elementos (desreferenciamento dentro de estruturas
  - Necessidade de percorrer a lista para acesso a um determinado elemento
- Uso
  - Inserção e remoção são operações mais frequentes
  - Não se conhece o tamanho máximo da lista

Lista Duplamente encadeada

Operação	Complexidade	Sobre
push_front	O(1)	Envolve apenas manipulação de alguns ponteiros (possui o ponteiro para o início *begin*)
push_back	O(1)	Envolve apenas manipulação de alguns ponteiros (possui o ponteiro para o final *end*)
ordenada	O(N)	No pior caso envolve percorrer toda a lista
pop_front	O(1)	Envolve apenas manipulação de ponteiros da cabeça da lista
pop_back()	O(1)	Envolve apenas manipulação de ponteiros do final da lista
busca	O(N)	No pior caso temos que percorrer toda a lista para encontrar o elemento
Acesso (pos)	O(N)	No pior caso o elemento está no final

Lista Duplamente encadeada

- Vantagens
  - Melhor utilização da memória
  - Não é preciso definir previamente o tamanho da lista
  - Inserção e remoção sem necessidade de movimentar elementos
- Desvantagens
  - Acesso indireto aos elementos (desreferenciamento dentro de estruturas
  - Necessidade de percorrer a lista para acesso a um determinado elemento
- Uso
  - Inserção e remoção são operações mais frequentes
  - Não se conhece o tamanho máximo da lista
  - Necessidade de percorrer a lista em diferentes ordens (frente e reverso)

Filas e Pilhas

- As operações em filas e pilhas envolvem apenas partes das operações já presentes em listas. Assim, inserção, remoção e consulta são todas O(1)
- Não temos nenhuma operação O(N) pois não fazemos consultas que percorrem todas essas estruturas
  - Uma exceção seria o free_stack e free_queue dinâmicos que envolvem percorrer todos os elementos para liberar a memória, sendo então O(N)