

Lista de exercícios No. 2

1. Deseja-se construir uma árvore binária de pesquisa completa a partir de um vetor ordenado de tamanho $2^n - 1$. Escreva um algoritmo recursivo e um não recursivo, que insiram os elementos deste vetor, em uma árvore originalmente vazia, em uma ordem tal que a árvore resultante fique completa sem necessidade de balanceamento. Por exemplo, para o vetor 1-2-3-4-5-6-7, uma solução seria inserir os elementos na seguinte ordem: 4-2-1-3-6-5-7.

```
Arvore(1,n,v,raiz)
Arvore(esq,dir,v,raiz)
    Pos=(dir+esq)/2
    Insere(raiz,v[pos]) //faz h comparações, h=altura de inserção do nodo
    Se esq<dir então
        Arvore(esq,pos-1,v,raiz)
        Arvore(pos+1,dir,v,raiz)
    Fim se

Arvore(esq,dir,v,raiz)
    delta=n+1
    Para i=1 ate lg(n+1)
        Para j=delta/2 ate n passo delta
            Insere(raiz,v[j]) //faz h comparações, h=altura de inserção do nodo
        delta=delta/2
    Fim para
```

2. Proponha um algoritmo para a solução do problema das 8 rainhas utilizando backtracking.

```
NRainhas(1,Pos,n);

NRainhas(c,Pos,n)
Se c>n Imprima(Pos)
Senão Para i=1 até n
    Pos[c]=i
    Se Salva(Pos,n) NRainhas(c+1,Pos,n)
Fim para
```

3. Proponha um algoritmo para a solução do problema da soma de subconjuntos utilizando backtracking.

```
SSC(conj,subconj=vazio,v,valor,soma=0,i=1,n)

SSC(conj,subconj,v,valor,soma,i,n)
Se soma = valor então Imprima(subconj) e termine
Senão Se i>n então Imprima("sem solução") e termine
Senão Se soma+conj(i) <= valor então
    Insere(subconj,i)
    soma=soma+v[i]
    SSC(conj,subconj,v,valor,soma,i+1)
    Remove(subconj,i)
```

```
soma=soma-v[i]
Fim Se
SSC(conj,subconj,v,valor,soma,i+1)
Fim Se
```

4. Todo algoritmo recursivo pode ser transformado em um equivalente não recursivo. A eliminação da recursividade de um algoritmo pode ser considerada uma solução de compromisso? Quais as vantagens e desvantagens de se eliminar a recursividade? Justifique as respostas.

Sim, trocam-se tempo e memória por simplicidade.

5. Proponha um algoritmo para a solução do problema da mochila 0/1 utilizando branch-and-bound.

```
Mochila(i,w,v,ub,max)
Se i>N //todos os objetos foram inspecionados
  Guarda solução
  max=v
senão
  Se w+peso[i]<=W //objeto i cabe na mochila
    Coloca i na mochila
    ub=v+valor[i]+(W-w-peso[i])*valor_peso[i+1]
    se ub>max Mochila(i+1,w+peso[i],v+valor[i],ub,max) //critério de poda
    Remove i da mochila
  Fim se
  ub=v+(W-w)*valor_peso[i+1] // não coloca o objeto
  se ub>max Mochila(i+1,w,v,ub,max) //critério de poda
Fim
```

```
Mochila(i=1,w=0,v=0,ub=0,max=0)
```

6. Proponha uma heurística gulosa para o problema da linha de montagem.

A cada etapa da linha, escolher a esteira que tiver o menor custo de produção somado ao custo de transferência.

7. Compare a heurística proposta no exercício anterior com a versão por programação dinâmica quanto ao tempo de execução, gasto de memória e eficácia (qualidade da solução obtida). Execute a heurística para o exemplo do livro e compare os resultados. A heurística produziu um resultado ótimo?

Mesma ordem de complexidade de tempo de execução (linear), gasto de memória menor, ótimo não é garantido. Solução para a heurística: 121111, custo 39.

8. Proponha uma heurística gulosa para o problema da mochila 0/1.

Colocar os objetos ordenados por valor/peso até não haver mais espaço para os próximos.

9. Proponha um algoritmo para a solução exata do problema do caixeiro viajante utilizando a técnica de branch-and-bound.

```
CV (cidades, i, min, custo)
Se i=N então //última cidade
    Se custo+c[i,1]<min
        Guarde solução
        min=custo+c[i,1]
    Fim se
senão para k:=1 até N-i+1 faça
    Se custo+c[cidades[i],cidades[i+1]]) <min //critério de poda
        CV(cidades,i+1,min,custo+c[cidades[i],cidades[i+1]]) //prox
    Rotaciona (cidades,i)
Fim para
Fim
```

```
Programa Principal
    min=0; custo=0
    CV(cidades,2,min,custo) //avalia segunda cidade em diante
Fim.
```

10. Proponha um algoritmo para a solução do problema de bin packing utilizando força-bruta.
11. Proponha um algoritmo para a solução do problema de bin packing utilizando branch-and-bound.
12. Proponha um algoritmo para a solução aproximada do problema de bin packing em tempo polinomial.
13. Uma heurística pode ser considerada uma solução de compromisso? Justifique.

Sim, troca-se tempo de execução pela qualidade da solução.

14. Você precisa solucionar um determinado problema de otimização. Escreva um roteiro que o auxilie a escolher a melhor técnica de projeto para a solução do problema, com base nas suas características particulares. O roteiro deve ser da forma: Se o problema tem o conjunto de características A, então a técnica é X, senão, se o problema tem o conjunto de características B, então a técnica é X, senão ... etc. Logicamente, as técnicas mais eficientes devem ser as primeiras opções a serem avaliadas.

Se o problema pertence a P

Se o problema possui a propriedade da escolha gulosa use um algoritmo guloso
Senão se o problema possui a propriedade da subestrutura ótima e sobreposição de subproblemas use programação dinâmica
Senão use força-bruta

Senão se for necessário obter o ótimo

Se for possível encontrar um critério de poda eficiente use branch-and-bound
Senão se o problema possui a propriedade da subestrutura ótima e sobreposição de subproblemas use programação dinâmica
Senão use força-bruta

Senão use uma heurística gulosa

15. Escreva um algoritmo não determinístico $O(1)$ para decidir se um vetor contém elementos repetidos.

16. O problema de otimização do Caixeiro Viajante pertence a NP? Justifique.

Não, ele é NP-difícil.

17. O problema de decisão do Caixeiro Viajante pertence a NP? Justifique.

Sim, ele é NP-completo.

18. O que podemos afirmar sobre um problema de decisão A se encontrarmos uma solução polinomial em máquina determinista para ele? Justifique.

Ele pertence a P.

19. Diga se a seguinte afirmativa é verdadeira ou falsa e justifique. Se eu tenho um novo problema de decisão A pertencente a NP, para eu provar que A está em NP-Completo basta encontrar uma redução polinomial de algum outro problema NP-Completo para A.

Verdadeiro, pois Cook provou que SAT é NP-completo e se reduz aos demais problemas em NP-completo.

20. Seja A um problema pertencente a NP localizado no diagrama abaixo. Se alguém conseguir provar que é possível resolver o problema A em tempo polinomial em uma máquina determinista o que poderemos dizer sobre as classes P, NP e NP-Completo? Serão iguais.