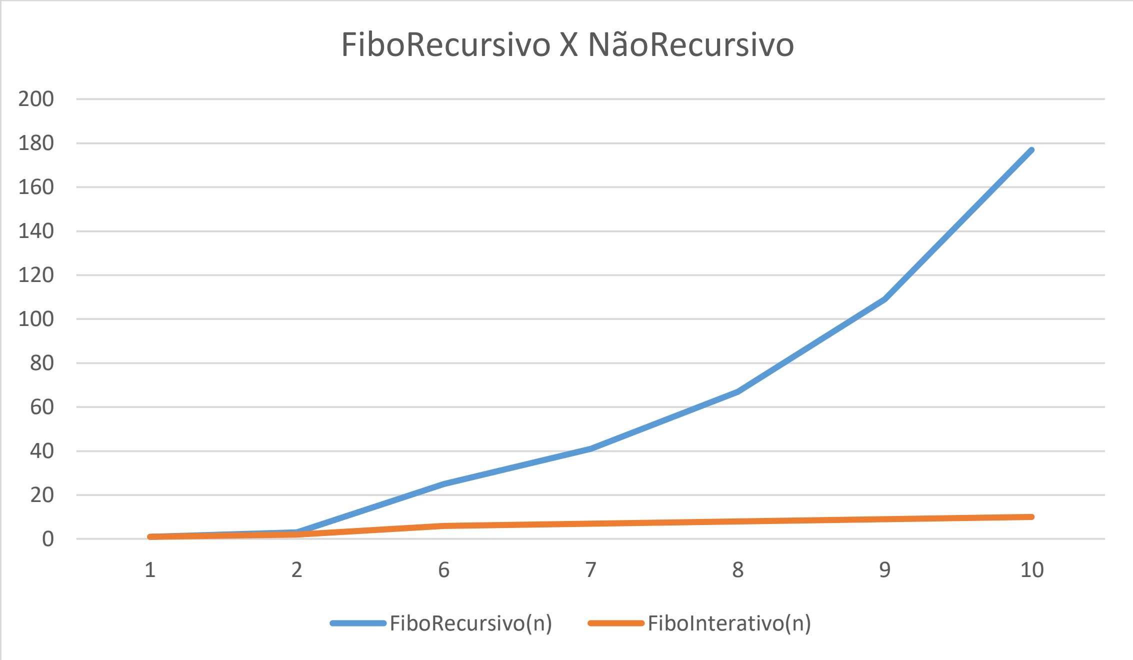
Aluno: Jean Bertrand Paixão da Silva - 1201224412

Respostas

Questao 1

a) Para o referido problema foi implementado um codigo em C utilizando a IDE "DEV – C++" foi feito uma função recursiva que obteve complexidade de $O(2^n)$ e uma função não recursiva de complexidade O(n). após escrever as duas funçoes fiz o teste com valores de entrada de 1 a 40 e pude observar que quanto maior o numero a função recursiva demora mais por executar bem mais chamadas do que a função não recursiva como podemos observar no grafico a seguir.



Como exemplo usei apenas o intervalo de 1 a 10 para melhor visualizar a quantidade de chamadas executadas. Enquanto o fiboInterativo se mantem mais linear o recursivo tem uma curva bem maior.(a esquerda os valores de y são as quantidades de chamadas que o programa executa para encontrar o e-nesimo termo).
Para executar o programa na IDE utilizada basta abrir o codigo e clicar em "Execute – Compile & Run", o programa já roda ambos os codigos mostrando a quantidade de chamadas e o e-nesimo termo da sequencia Fribonacci. Onde "FiboRecursivo(n)" representa a função recursiva , "m chamada" a quantidade que o programa foi executado e "y" após a virgula é o e-nésimo termo da sequencia fibonacci.

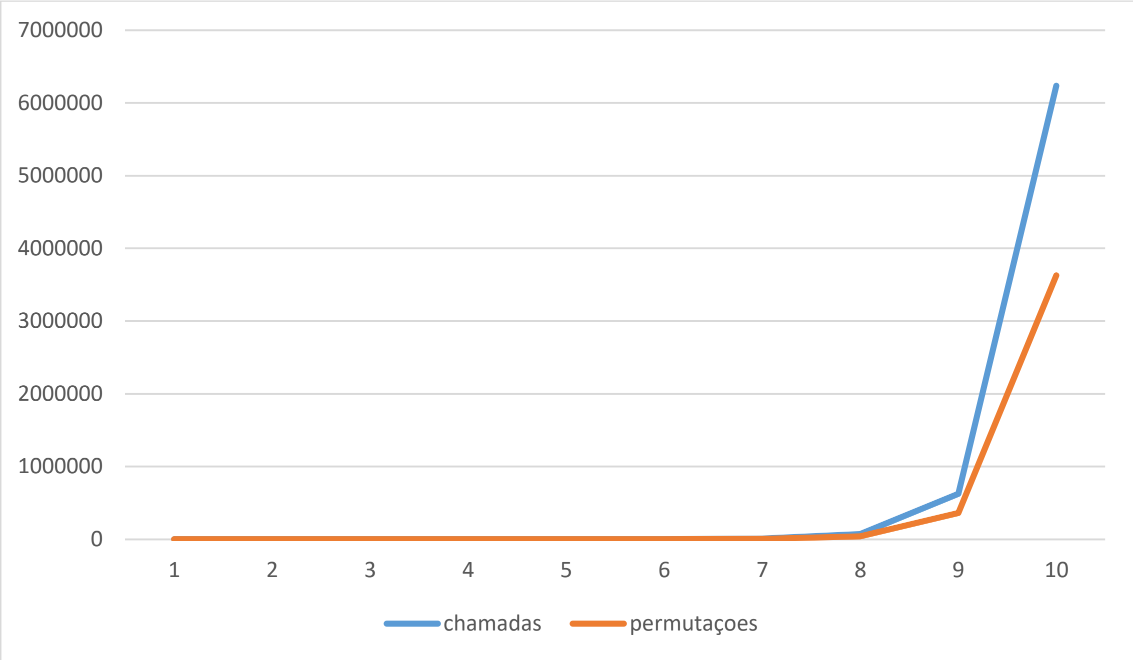
C:\Users\jean_\Documents\Jean Bertrand\UFRR\analise de algoritmo\lista_2.exe

```
FiboRecursivo(1), 1 chamada, 1
FiboInterativo(1),1 chamada, 1
FiboRecursivo(2), 3 chamada, 1
FiboInterativo(2),2 chamada, 1
FiboRecursivo(3), 5 chamada, 2
FiboInterativo(3),3 chamada, 2
FiboRecursivo(4), 9 chamada, 3
FiboInterativo(4),4 chamada, 3
FiboRecursivo(5), 15 chamada, 5
FiboInterativo(5),5 chamada, 5
FiboRecursivo(6), 25 chamada, 8
FiboInterativo(6),6 chamada, 8
FiboRecursivo(7), 41 chamada, 13
FiboInterativo(7),7 chamada, 13
FiboRecursivo(8), 67 chamada, 21
FiboInterativo(8),8 chamada, 21
FiboRecursivo(9), 109 chamada, 34
FiboInterativo(9),9 chamada, 34
FiboRecursivo(10), 177 chamada, 55
FiboInterativo(10),10 chamada, 55
FiboRecursivo(11), 287 chamada, 89
FiboInterativo(11),11 chamada, 89
FiboRecursivo(12), 465 chamada, 144
FiboInterativo(12),12 chamada, 144
FiboRecursivo(13), 753 chamada, 233
FiboInterativo(13),13 chamada, 233
FiboRecursivo(14), 1219 chamada, 377
FiboInterativo(14),14 chamada, 377
FiboRecursivo(15), 1973 chamada, 610
FiboInterativo(15),15 chamada, 610
```

Exemplo entrada e saida

A conclusão seria que o melhor caso para ambos é quando solicitamos n = 0 ou n = 1 pois o custo seria apenas 1. E o pior caso seria quando n é um valor muito grande. Sendo ainda que o recursivo é muito pior do que o não recursivo.
O codigo pode ser acessado em "https://github.com/shaolinbertrand/JeanBertrand_AA_Lista2_rr_2019./blob/master/FiboRecursivoXFiboInterativo"

b) Para este problema foi feito um algoritmo em C na IDE "DEV-C++", a ideia é o usuario digitar o tamanho do numero que ele deseja digitar e em seguida digitar o numero que é armazzenado em um vetor e depois disso é gerado todas as permutações para o determinado numero. É importante levar em consideração que o usuario deve entrar com numeros distintos.
após fazer algumas pesquisas constatei que dado um arranjo de tamanho n a quantidade de permutações geradas é de n!. ou seja para n = 4 temos um total de 24 permutações. O programa recursivo por tanto é chamado tem complexidade O(n!).
fiz o teste com numeros de tamanho 1 até 10 e tive os seguintes resutados:



Numero de prmutações e chamadas recursivas para valores de 1 a 10

Logo podemos constatar que o melhor caso seria n = 1 pois gera apenas uma permutação e uma chamada da função recursiva. E quanto maior o valor de n pior fica o programa pois para n = 10 gera 10! Permutações e 2 * 10! Chamadas da função recursiva.

O programa após gerar todas as permutações mostra a quantidade de permutações e qauntas chamadas foram realizadas.

C:\Users\jean_\Documents\Jean Bertrand\UFRR\analise de algoritmo\lista_2.exe

```
digite o tamanho do numero que deseja fazer a permuta
4
digite o valor 1: 1
digite o valor 2: 2
digite o valor 3: 3
digite o valor 4: 4
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 3 2
1 4 2 3
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 3 1
2 4 1 3
3 2 1 4
3 2 4 1
3 1 2 4
3 1 4 2
3 4 1 2
3 4 2 1
4 2 3 1
4 2 1 3
4 3 2 1
4 3 1 2
4 1 3 2
4 1 2 3
24 permutas realizadas
funcao recursiva chamada 41 vezes
--------------------------------
Process exited after 5.006 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exemplo entrada e saida para um numero de tamanho 4

Infelizmente não conseguir implementar um codigo que resolvesse o problema de forma não recursiva, pensei em algumas soluçoes mas na hora de colocar em prática não obtve sucesso. Pois muitas vezes ele acabava repetindo as mesmas saidas logo não coloquei o codigo neste trabalho.

Codigo disponivel em
(“https://github.com/shaolinbertrand/JeanBertrand_AA_Lista2_rr_2019./blob/master/Permuta%C3%A7%C3%B5esRecursiva”)

2)

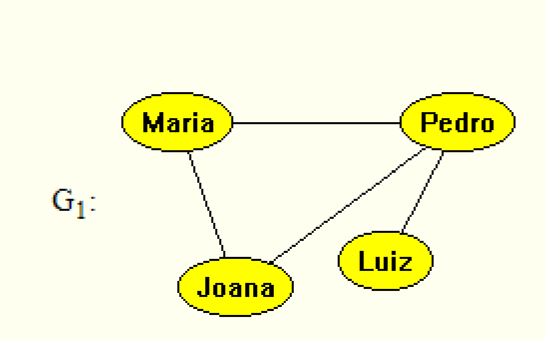
a) Um grafo *G(V,A)* é definido pelo par de conjuntos *V* e *A*, onde:

V - conjunto não vazio: os **vértices** ou **nodos** do grafo;
A - conjunto de pares ordenados *a*=(v,w), v e w ∈ V: as **arestas** do grafo.

Seja, por exemplo, o grafo G(V,A) dado por:

V = { p | p é uma pessoa }
A = { (v,w) | < v é amigo de w > }

Esta definição representa toda uma família de grafos. Um exemplo de elemento desta família (ver G_1) é dado por:
V = { Maria, Pedro, Joana, Luiz }
A = { (Maria, Pedro), (Pedro, Maria), (Joana, Maria), (Maria, Joana), (Pedro, Luiz), (Luiz, Pedro), (Joana, Pedro) , (Pedro, Joana) }



Fonte("http://www.inf.ufsc.br/grafos/definicoes/definicao.html")

c) GRAFO CONEXO
Um grafo G(V,A) é dito ser conexo se há pelo menos uma cadeia ligando cada par de vértices deste grafo G.
Fonte("http://www.inf.ufsc.br/grafos/definicoes/definicao.html")

CICLO

um ciclo em um grafo é um caminho fechado sem vértices repetidos. Mais precisamente, um ciclo (= cycle) é um caminho (v0, a1, v1, a2, v2, . . . , ak, vk) com k > 1 onde

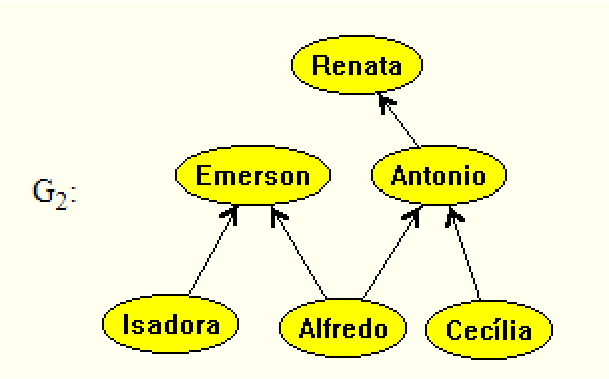
vk = v0 mas v0, v1, v2, . . . , vk–1 são distintos dois a dois.
Fonte("https://www.ime.usp.br/~pf/algoritmos_em_grafos/aulas/dag.html")

ACÍCLICO
Um grafo é acíclico (= acyclic) se não tem ciclos. Grafos acíclicos são conhecidos pelas iniciais DAG da expressão directed acyclic graph
Fonte("https://www.ime.usp.br/~pf/algoritmos_em_grafos/aulas/dag.html")

c) Em um grafo simples dois vértices v e w são adjacentes (ou vizinhos) se há uma aresta a=(v,w) em G. Está aresta é dita ser incidente a ambos, v e w. No caso do grafo ser dirigido, a adjacência (vizinhança) é especializada em:

Sucessor: um vértice w é sucessor de v se há um arco que parte de v e chega em w. Em G2, por exemplo, diz-se que Emerson e Antonio são sucessores de Alfredo.

Antecessor: um vértice v é antecessor de w se há um arco que parte de v e chega em w. Em G2, por exemplo, diz-se que Alfredo e Cecília são antecessores de Antonio.

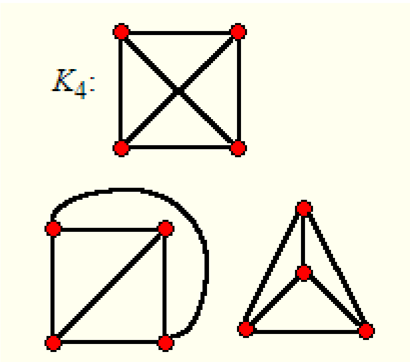


Fonte("http://www.inf.ufsc.br/grafos/definicoes/definicao.html")

d)

Um grafo G(V,A) é dito ser planar quando existe alguma forma de se dispor seus vértices em um plano de tal modo que nenhum par de arestas se cruze.

Há baixo aparecem três representações gráficas distintas para uma K4 (grafo completo de ordem 4). Apesar de haver um cruzamento de arestas na primeira das representações gráficas, a K4 é um grafo planar pois admite pelo menos uma representação num plano sem que haja cruzamento de arestas (duas possíveis representações aparecem nas figuras ao lado).



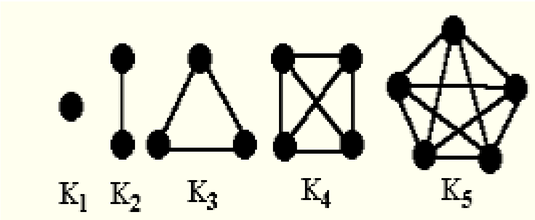
Fonte("http://www.inf.ufsc.br/grafos/definicoes/definicao.html")

e)

GRAFO COMPLETO

Um grafo é dito ser completo quando há uma aresta entre cada par de seus vértices. Estes grafos são designados por Kn, onde n é a ordem do grafo.

Um grafo Kn possui o número máximo possível de arestas para um dados n. Ele é, também regular-(n-1) pois todos os seus vértices tem grau n-1.



GRAFO BIPARTIDO

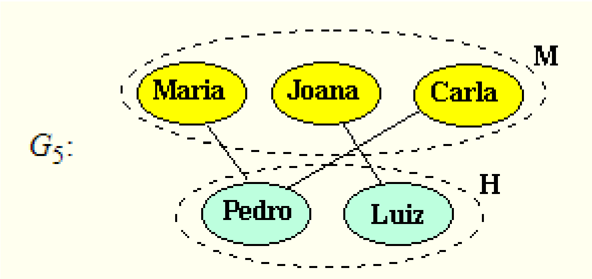
Um grafo é dito ser bipartido quando seu conjunto de vértices V puder ser particionado em dois subconjuntos V1 e V2, tais que toda aresta de G une um vértice de V1 a outro de V2.

Para exemplificar, sejam os conjuntos H={h | h é um homem} e M={m | m é um mulher} e o grafo G(V,A) (ver o exemplo G5) onde:

V = H U M

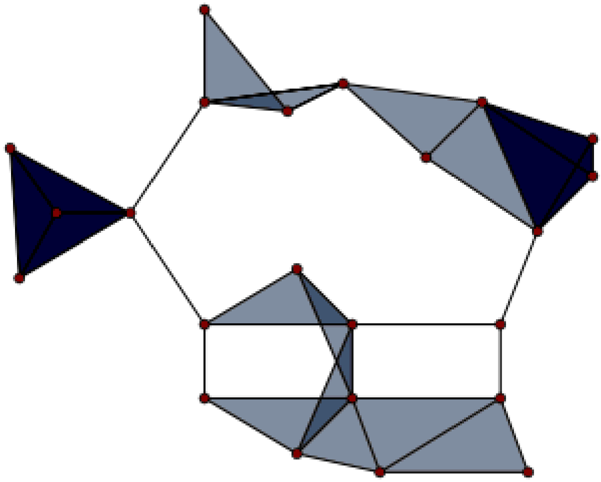
A = {(v,w) | (v ∈ H e w ∈ M) ou (v ∈ M e w ∈ H)

e <v foi namorado de w>}



Clique - Em um grafo não-orientado é um subconjunto de seus vértices tais que cada dois vértices do subconjunto são conectados por uma aresta.

Exemplo:



f)

DIGRAFO (Grafo Orientado)

Considere, agora, o grafo definido por:

V = { p | p é uma pessoa da família Castro }

A = { (v,w) | < v é pai/mãe de w > }

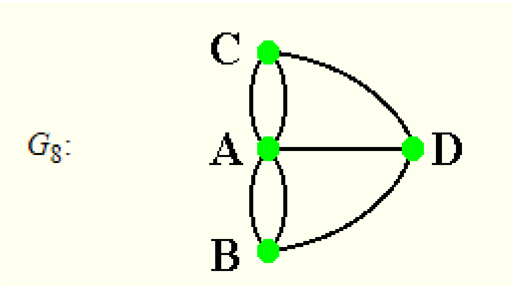
Um exemplo de deste grafo (ver G2) é:

V = { Emerson, Isadora, Renata, Antonio, Cecília, Alfredo }

A = {(Isadora, Emerson), (Antonio, Renata), (Alfredo, Emerson), (Cecília, Antonio), (Alfredo, Antonio)}

MULTIGRAFO

Um grafo G(V,A) é dito ser um multigrafo quando existem múltiplas arestas entre pares de vértices de G. No grafo G8, por exemplo, há duas arestas entre os vértices A e C e entre os vértices A e B, caracterizando-o como um multigrafo.



3)

Matriz de incidência – Uma matriz de incidência representa computacionalmente um grafo através de uma matriz bidimensional, onde uma das dimensões são vértices e a outra dimensão são arestas.

Matriz de Adjacência – Dado um grafo G, a matriz de adjacências r = (r_{ij}) é uma matriz n x n tal que:

n – numero de vértices

r_{ij} = 1 se(v_i,v_j) pertence a E

r_{ij} = 0

Vantagem – Tempo de acesso mais rápido.

Desvantagem – Maior espaço para armazenamento

Lista de adjacência – É um conjunto de n listas A(v),uma para cada vértice v e contem vértices w adjacentes a v em g.

Vantagem – Menor espaço para armazenamento

Desvantagem – Tempo de acesso mais lento

Fonte("http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/Grafos/RepImpl/rep_impl.html")

4)

a)Enumeração Explicita x Implicita - enumeração explicita é uma técnica de força bruta,ou seja ele soluciona o problema mas nem sempre da a melhor solução,ja o algoritmo de enumeração implicita busca a solução mais eficiente possível. Fonte("http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-75901969000400001")

b)Programação Dinamica - Programação dinâmica é um método para a construção de algoritmos para a resolução de problemas computacionais, em especial os de otimização combinatória.Exemplo: Multiplicação de Matrizes. Fonte("https://www.lume.ufrgs.br/bitstream/handle/10183/16007/000680180.pdf?sequence=1")

c) Algoritmo guloso - Um algoritmo guloso é míope: ele toma decisões com base nas informações disponíveis na iteração corrente, sem olhar as consequências que essas decisões terão no futuro. Um algoritmo guloso jamais se arrepende ou volta atrás: as escolhas que faz em cada iteração são definitivas. Exemplo: Algoritmo de Prim. Fonte("https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/guloso.html")

d)backtraking - É um algoritmo genérico que busca, por força bruta, soluções possíveis para problemas computacionais (tipicamente problemas de satisfações à restrições). Fonte("https://pt.stackoverflow.com/questions/103184/o-que-%C3%A9-um-algoritmo-backtracking/103670")

6)
a) SAT x NP-Completude

A classe dos problemas NP-difíceis contém os problemas de complexidade maior ou igual a do problema SATisfabilidade. A complexidade de SAT é um limite inferior para a complexidade de um problema P é fazer uma redução polinomial de SAT a este problema P.

Fonte("http://www.cin.ufpe.br/~katiag/cursos/pos_NoACESS/ComputCientifica_ModuloAlgoritmos/NocoesNPC.ppt")

b)

Classe P - Consiste nos problemas que podem ser resolvidos em tempo Polinomial,são problemas que podem ser resolvidos no tempo $O(n^k)$.Exemplo:Encontrar um numero primo.

Classe NP – Consiste em problemas que são verificaveis em tempo polinomial.Exemplo: Ploblema do ciclo hamiltoniano.

Classe NP-Dificil – Um problema é NP-dificil se todos os problemas NP não são mais dificeis que ele. Exemplo:Problema da Parada.

Classe NP-Completo – Um problema é NP-completo se for NP-dificil e estiver em NP.Exemplo : Problema do Caixeiro Viajante.
Fonte("https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto.html")