Gabriel Barbosa da Silva

Profa. Ms. Seila Vasti Faria de Paiva

Bacharelado em Ciência da Computação

Disciplina: COA – Complexidade de Algoritmos

Pesquisa: Paradigmas de Projeto de Algoritmos

Técnica de projeto de algoritmos por indução(Divisão e Conquista)

Definição - Esse tipo de algoritmo chama a si mesmo várias vezes, recursivamente, para resolver um problema decompondo-o em subproblemas menores. Eles seguem a abordagem de dividir e conquistar, que seguem três passos em cada nível de recursão, sendo eles: a divisão de um problema em um determinado número de subproblemas, conquistar os subproblemas recursivamente, então, combinar as soluções encontradas nos subproblemas para solucionar o problema original.

Para algoritmos recursivos, podemos determinar seu tempo de execução por uma *equação de recorrência* , que descreve o tempo de execução global sobre um problema de tamanho *n*. Uma recorrência é descrita para um caso inicial k e uma fórmula recursiva para k+1, por exemplo:

$$F(n) = \begin{cases} 1 & \text{se } n = 0 \\ n.F(n-1) & \text{se } n \geq 1 \end{cases}$$

A partir dessa recorrência obtemos a fórmula fechada, que para algoritmos é o tempo de execução para uma determinada entrada *n*.

$$\begin{aligned} F(n) &= n.F(n-1) \\ &= n.(n-1).F(n-2) = \cdots = \\ &= n.(n-1).(n-2)\ldots 2.1.1 \\ &= n! \end{aligned}$$

Conseguimos resolver recorrências por Indução Matemática, mas para acharmos os limites assintóticos de um algoritmo, disponibilizamos três métodos. O primeiro é o *método da substituição*, onde assumimos um limite hipotético e usamos a indução matemática para provar que nossa suposição era a correta. A segunda é a da *árvore de recursão*, que transforma a recorrência em uma árvore e cada nó tem o custo por nível de chamada. E por último, temos o *método mestre*, que fornece limites para recorrências na forma

$$T(n) = aT(n/b) + f(n)$$

Sendo $a \geq 1$, $b > 1$ *e* $f(n)$ *uma função dada;*

E a comparamos com três casos, desde que a recorrência respeite a forma de um dos casos.

Aplicabilidade - Umas das aplicações da abordagem de *divisão e conquista* é no paralelismo entre várias CPUs, podemos decompor um processo em vários para obter o melhor desempenho em um sistema multiprocessado. Isso é perfeito para operações entre matrizes, pois assim podemos realizar uma multiplicação dividindo-a em subproblemas de multiplicação e adição, para depois obter o resultado da matriz completa.

Algoritmo de exemplo - Como exemplo de *divisão e conquista*, utilizei o algoritmo de A.A. Karatsuba, que serve para multiplicar números naturais muito grandes e tem complexidade $O(n^{1.584})$.

	milissegundos		
n	ordinário	Karatsuba	
8	0.000	0.008	–
16	0.002	0.010	500%
32	0.007	0.019	271%
64	0.027	0.045	167%
128	0.101	0.128	127%
256	0.388	0.368	95%
512	1.541	1.111	72%
1024	6.079	3.400	56%
2048	24.460	12.000	49%

Algoritmos Gulosos

Definição - Algoritmos gulosos(ou gananciosos), seguem uma abordagem de escolher a solução que ofereça os melhores benefícios naquele instante, ou seja, é um paradigma que não vê as soluções passadas e as futuras para prever uma melhor. Isso o torna ótimo em certos problemas, como o de troco para moedas, em que temos que escolher o mínimo de notas e moedas possíveis como troco. Para isso, temos que ter uma visão top-down, em que o programa pode escolher as maiores notas para solucionar o problema. Mas temos que ficar atentos para impor algumas condições, imagine que temos notas de 100, 50,25,10,1 e temos que entregar um troco de 30. Nesse caso o algoritmo vai pegar a maior nota que seria 25 e depois mais cinco notas de 1 mas isso está errado, pois podemos entregar três notas de 10. Portanto, é preciso identificar se o programa funciona para todos os casos.

Aplicabilidade - Um outro uso destes algoritmos, seria em conjunto com algoritmos de roteamento dinâmicos. Tais algoritmos re-calculam as rotas entre os roteadores da rede sempre que há necessidade. Por exemplo, quando o tráfego aumenta muito entre dois

roteadores (modificando os pesos das arestas do grafo da rede) ou quando uma falha física elimina uma ou mais rotas (modificando o grafo da rede). Nesses casos, a árvore de menor custo indicaria a melhor rota que conecta todos os roteadores. "Melhor", neste caso leva em consideração a velocidade ou o throughput da rede.

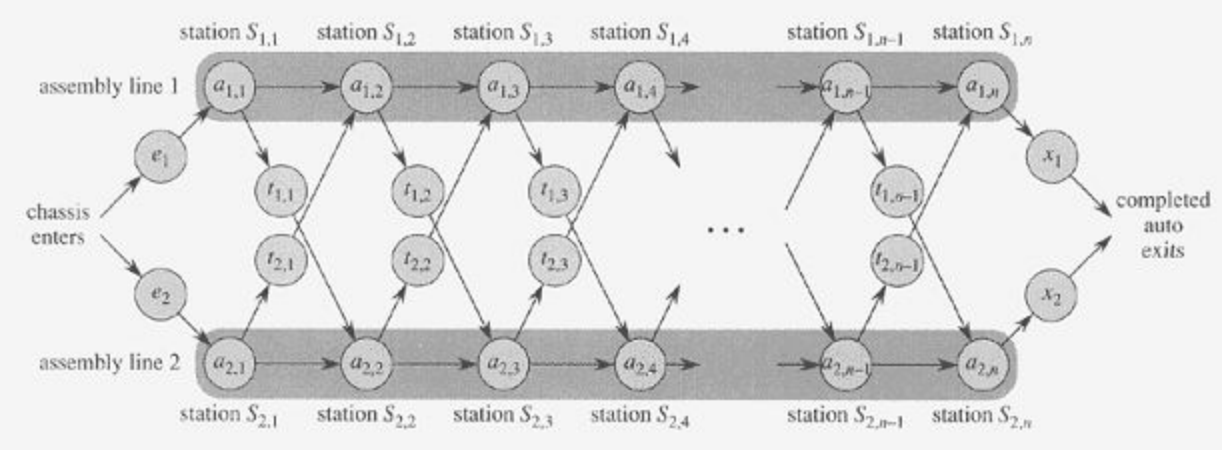
Algoritmo de exemplo - Para exemplo, escolhi um algoritmo que determina o menor custo para passar por todas as cidades de uma região, dada uma matriz Z de $[k][j]$ posições, onde o valor representa o custo para ir da cidade *k* até a *j*. Esse algoritmo utiliza o algoritmo da árvore de espalhamento mínimo.

Programação Dinâmica

Definição - Em contraste com o paradigma de *divisão e conquista*, o método de *programação dinâmica* guarda os resultados de subproblemas já calculados para economizar tempo. Diferente dos outros algoritmos já vistos, a *programação dinâmica* é aplicada em problemas de otimização. Nestes problemas podemos obter muitas soluções possíveis, mas queremos somente uma solução ótima, em vista que podemos encontrar várias soluções ótimas.

Aplicabilidade - Umas das aplicações desse paradigma é na programação em uma linha de montagem em um fábrica. Por exemplo, imagine duas linhas de montagem com J estações. Ao entrar uma peça, ela passa por cada estação de forma a nunca repetir um tipo de estação e a peça vai sempre entrar na estação que vai levar o menor tempo para montar a peça anterior.

Então vemos que nesse caso, queremos sempre minimizar o tempo de montagem de uma peça e obter o menor tempo desde da entrada da peça na primeira estação até a última estação.



Algoritmo de exemplo - Como exemplo, escolhi o algoritmo de Maior subsequência Comum. Dadas duas sequências, encontre o comprimento da maior subsequência presente em ambas. Uma subsequência é uma sequência que aparece na mesma ordem relativa, mas não necessariamente contígua. Por exemplo, “abc”, “abg”, “bdf”, “aeg”, ”acefg”, .. etc. são subsequências de “abcdefg”.

Trabalhos citados

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos: Teoria e Prática. 2. ed. Rio de Janeiro: Editora Campus, 2002.

Site: https://www.geeksforgeeks.org/divide-and-conquer/; Visitado em: 28/12/2020.

Site: https://www.geeksforgeeks.org/karatsuba-algorithm-for-fast-multiplication-using-divide-and-conquer-algorithm/; Visitado em: 28/12/2020.

A. G. Silva(maio,2018). Projeto e Análise de Algoritmos. Página 34. Disponível em: https://www.inf.ufsc.br/~alexandre.goncalves.silva/courses/18s1/ine410104/slides/aula0504.pdf

Site: https://www.geeksforgeeks.org/greedy-algorithms/; Visitado em: 29/12/2020.

Site: https://www.geeksforgeeks.org/minimum-cost-connect-cities/; Visitado em: 30/12/2020.

Site: https://www.geeksforgeeks.org/dynamic-programming/; Visitado em: 30/12/2020.

Site: https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/; Visitado em: 30/12/2020.