Relatório 2º projecto ASA 2020/2021

Grupo: al102

Aluno(s): Francisco Ribeiro (95578) e Diogo Lopes (96732)

Descrição do Problema e da Solução

Este projeto tem como objetivo descobrir a melhor distribuição (de menor custo total) de *N* processos por dois processadores *X* e *Y*, sabendo que os custos de comunicação entre processos existem quando estes se encontram em processadores diferentes.

Para resolver o problema decidimos construir um grafo, em que os vértices nos extremos são os processadores $\{X, Y\}$ e os intermédios são os processos $\{p_1, \ldots, p_N\}$. Os arcos do grafo correspondem $\{(X, p_i), (p_i, Y), \forall_{1 \le i \le N}\}$ e os arcos bidirecionais de comunicação entre processos, *K.* A solução deste problema reside no cálculo do fluxo máximo do grafo gerado. De forma a calcular o fluxo máximo decidimos recorrer ao algoritmo de Dinic, que é baseado em caminhos de aumento e tem complexidade assintótica inferior a Edmonds-Karp, uma vez que para este problema, $E \le O(V^2)$. Ao contrário de algoritmos de pré-fluxo, o algoritmo utilizado respeita a conservação do fluxo, algo necessário dado que existem arcos bidirecionais.

Fontes: https://www.geeksforgeeks.org/dinics-algorithm-maximum-flow/
https://www.arl.wustl.edu/~jst/cse/542/text/sec17.pdf
https://www.cs.cmu.edu/~15451-f18/lectures/lec11-flow2.pdf

Análise Teórica

```
sendFlow(s, f, t, L)
  assert(s != t)

  for i=1; L[i] <= G.Adj[s].size(); i++              (1)
    Let Edge be (s, L[i])

    if L[i].level == u.level+1 && Edge.f < Edge.c && !Edge.deadEnd
      CurrentFlow := min(f, Edge.capacity - Edge.flow)
      TempFlow := sendFlow(L[i], CurrentFlow, t, L)      (2)

      if TempFlow > 0
        Let ReverseEdge be (L[i], s)
        Edge.f += TempFlow
        ReverseEdge.f -= TempFlow
        return TempFlow

      else Edge.deadEnd = 1   //Correção do algoritmo (5)

  return SUCCESS
```

```
DinicMaxFlow(s, t)
  Let total be the return value for max flow
  Let INF be the infinite
  assert(s != t)

  //t é alcançável por s, 'levels' são atualizados
  while BFS(s, t)                                    (3)
    L := G.V          //Lista de vértices a visitar

    while flow := sendFlow(s, INF, t, L)             (4)
      total += flow

  return total
```

Análise teórica da complexidade total e das várias etapas da solução proposta:

Para este problema: $|V| = N + 2, |E| = 2N + 2K, onde\ K \le \sum_{i=1}^{N-1} i = \frac{(N-1)*N}{2} = O(N^2)$

- Leitura dos dados de entrada + construção do grafo: $O(N + K) = O(N^2)$
- Enviar fluxo por caminhos de aumento na pesquisa em profundidade, obtemos uma complexidade de $O(V * E)$: executamos o ciclo (1) pelas adjacências, que juntamente com o ciclo (2), o algoritmo percorre no máximo todos os vértices e arcos.

Relatório 2º projecto ASA 2020/2021

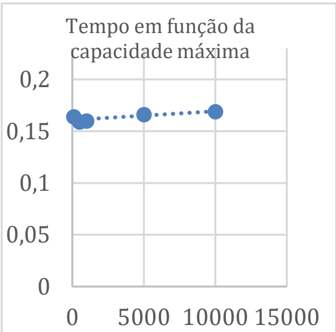
Grupo: al102

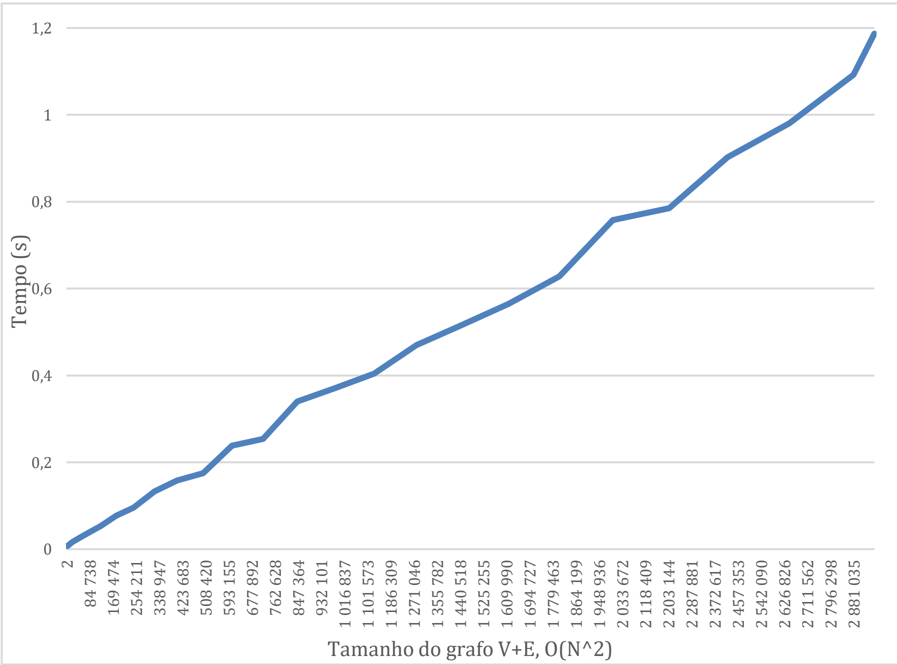
Aluno(s): Francisco Ribeiro (95578) e Diogo Lopes (96732)

- Correção do algoritmo (5): quando fazemos *backtracking* (2), verificamos se foi enviado fluxo pelo arco a analisar. Caso não tenha sido, marcamo-lo como "morto" tendo esta ação um custo temporal de $O(\ 1\ )$. Não melhora a complexidade assintótica do algoritmo, mas é uma melhoria que podemos introduzir.
- Encontrar o fluxo máximo: fazendo uso do algoritmo de Dinic, recorremos à BFS, que percorre cada vértice e cada arco apenas uma vez tendo complexidade temporal de $O(V + E) = O(E)$. Após este passo, calculamos os caminhos de aumento referidos anteriormente em tempo $O(V * E)$ (4). A cada iteração do ciclo (3) sabemos que, pelo menos, a altura de um vértice muda (BFS), o que leva a que o número de iterações seja $O(V)$. Complexidade do algoritmo de Dinic: $O\big(V * (E + VE)\big) = O(V^2E)$.
- Complexidade total: $O(V^2E) = O(N^4)$.

Avaliação Experimental dos Resultados

Para testar o programa, criámos uma série de grafos usando o algoritmo fornecido pelo professor, *gen2procs*, com 10000 capacidade máxima (constante) e número de processos entre 2 e 2500. A *seed* usada foi "1".





Por observação do gráfico da direita, numa primeira análise, reparamos que o tempo tem uma relação linear com V+E, pelo que é linear perante $O(N^2)$, logo tem uma relação $O(N^3)$. Mas, como o K pode variar entre 0 e $\frac{(N-1)*N}{2}$, quanto mais próximo estiver de 0, mais linear será a função e do último valor, mais quadrática. Podemos reparar que o gráfico não é totalmente linear, o que se deve ao K ter valores baixos (provavelmente devido à *seed* escolhida). Assim a função fica com complexidade assintótica de $O(N^4)$, o que condiz com a análise teórica.