ANÁLISE DO QUICK-UNION COM A HEURÍSTICA DOS TAMANHOS

Considere o algoritmo `quick-union` com a heurística dos tamanhos (implementação 3, aula 2):

```
 1. #include <stdio.h>
 2. #define N 1000
 3. main() {
 4.   int i, j, p, q, t, id[N], tam[N];
 5.
 6.   for (i=0; i < N; i++) { id[i] = i; tam[i] = 1; }
 7.   while (scanf("%d %d", &p, &q) == 2) {
 8.     for (i=p; id[i]!=i; i=id[i]);
 9.     for (j=q; id[j]!=j; j=id[j]);
10.     if (i != j) {
11.       if (tam[i] > tam[j]) { id[j] = i; tam[i] += tam[j]; }
12.       else { id[i] = j; tam[j] += tam[i]; }
13.       printf("      %d %d\n", p, q);
14.     }
15.   }
16. }
```

Vamos mostrar que, no início de qualquer iteração do `while` da linha 7,

> qualquer que seja `s`, se `r` é o representante de `s` (ou seja, `r = i` após a execução da linha 8 com `p = s`), a linha 8 executa no máximo log `tam[r]` vezes a atualização `i = id[i]`.

Podemos provar este fato por indução no número de iterações do `while` da linha 7. Na primeira iteração de `while`, a atualização `i = id[i]` não é executada nenhuma vez já que `id[i]` = `i` para todo `i`. Portanto a afirmação vale pois `tam[i]` = `1` para todo `i` e $\log 1 = 0$. Suponha agora que a afirmação acima vale no início de uma iteração arbitrária do `while` da linha 7. Vamos mostrar que ela vale também no fim dessa iteração. Se `i = j`, não há nada a provar. Se `i != j`, considere que `tam[i]`>`tam[j]` (o outro caso é análogo). Neste caso, fizemos `id[j] = i`, ou seja, `i` passou a ser o representante de todos os que antes eram representados por `i` ou por `j`. Fora isso, aumentamos o valor de `tam[i]`. Para todo `s` que tem representante diferente de `j` na linha 10, é fácil ver que a afirmação continua válida na linha 14 (a linha 8 requer exatamente o mesmo número de atualizações `i = id[i]`). Se, por outro lado, `s` tem `j` como seu representante na linha 10, então a linha 8 passa a requerer no máximo $1 + \log$ `tam[j]` atualizações `i = id[i]`. Como $1 + \log$ `tam[j]` $= \log(2$ `tam[j]`$) \leq \log($`tam[i]+tam[j]`$)$, a afirmação vale.

Da afirmação, podemos concluir que cada execução da linha 8 requer no máximo $\log N$ operações. Idem para a linha 9, e o algoritmo no total executa $O(M \log N)$ operações, onde M é o número de pares e N o tamanho do universo.

Notação assintótica: entenda $O(f(n))$ como "no máximo $cf(n)$, para alguma constante positiva arbitrária c".