II Maratona de Programação do IFB
D – Colônia de Bactérias

Daniel Saad Nogueira Nunes

O problema **Colônia de Bactérias** consistia em, dado um número n, representando a população inicial das bactérias, determinar qual o menor custo energético para chegar a uma população de tamanho m. As regras são:

- Uma bactéria pode se duplicar individualmente ou morrer, com custo x.
- Todas as bactérias podem se duplicar, com custo y.

Podemos modelar a solução deste problema através de uma relação de recorrência:

$$T(i) = \begin{cases} 0, i = n \\ (n-i) \cdot x, i < n \\ \min\{T(i-1) + x, T(i/2) + y\}, & \text{se } x > n \text{ par} \\ \min\{T(i-1) + x, T((i+1)/2) + x + y\}, & \text{se } x > n \text{ ímpar} \end{cases}$$

Suponha que queremos calcular o menor custo energético para chegar a uma população de i bactérias.

- Se $i = n$, o custo é zero, nosso caso base. Ou seja, para sair de n e chegar a n não é necessário nenhum gasto energético.
- Se $i < n$, é preciso que $n - i$ bactérias morram, com custo $(n - i) \cdot x$.
- Se $i > n$ temos duas possibilidades:
  - i par: a melhor solução vem do mínimo entre a solução para população $i-1$ mais o custo para uma bactéria individual se duplicar, ou da solução para a população $i/2$ mais o custo para todas se duplicarem. Não vale a pena considerar o caso que uma bactéria deva morrer, pois certamente o custo será maior.
  - i ímpar: a melhor solução vem do mínimo entre a solução para população $i - 1$ mais o custo para uma bactéria individual se duplicar, ou da solução para a população $(i+1)/2$ mais o custo para todas se duplicarem mais o custo para uma morrer. O caso em que todas se duplicam e uma individualmente se duplica já está contemplada pela solução da população $i-1$, então não é preciso considerá-la.

Solução

De acordo com a discussão anterior, obtemos o Algoritmo 1, baseado inteiramente na relação de recorrência apresentada, mas utilizando programação dinâmica.

Algorithm 1: PD(n, m, x, y)

Input: n, m, x, y
Output: Menor custo para sair de n e chegar em m

$T[n] \leftarrow 0$
for($i \leftarrow n-1; i \geq 0; i--$)
  $T[i] \leftarrow (n-i) \cdot x$
for($i \leftarrow n+1; i \leq m; i++$)
  $T[i] \leftarrow T[i-1] + x$
  if($i \mod 2$)
    $T[i] \leftarrow \min(T[i], T[(i+1)/2] + x + y)$
  else
    $T[i] \leftarrow \min(T[i], T[(i)/2] + y)$
return $T[m]$

Alternativamente, a solução em `C++` é apresentada a seguir:

Algorithm 2: `colonia-de-bacterias.cpp`

```
int64_t solve(int64_t n, int64_t m,int64_t x,int64_t y){
    vector<int64_t> v(m+1,0);
    for(int64_t i=0;i<=n;i++){
        v[i] = (n-i)*x;
    }
    for(int i=n+1;i<=m; i++){
        v[i] = v[i-1]+x;
        if(i%2==0){
            v[i] = min(v[i],v[i/2]+y);
        }
        else{
            v[i] = min(v[i],v[(i+1)/2]+y+x);
        }
    }
    return v.back();
}
```

Complexidade

A solução tem tempo de pior caso $\Theta(m)$, pois qualquer entrada da tabela pode ser computada olhando para um número constante de subsoluções e também porque ao todo temos $\Theta(m)$ entradas.