Paradigmas de Resolução de Problemas

Busca Completa – *Meet in the middle*

Prof. Edson Alves – UnB/FGA

Sumário

Meet in the Middle

Definição

- *Meet in the middle* (encontro no meio) é uma técnica de busca completa que divide o espaço de busca em duas partes, de aproximadamente mesmo tamanho, e tenta combinar as soluções parciais de cada metade em uma solução para o problema
- Ela foi introduzida em 1974 por E. Horowitz e S. Sahni
- Esta técnica só é viável se existir uma maneira eficiente de combinar os resultados de ambas metades
- Se existir, será mais eficiente fazer duas buscas em ambas metades do que uma única busca em todo o espaço
- Se o problema original tem complexidade $O(2^n)$, esta técnica tem o potencial de reduzir a complexidade para $O(2^{n/2})$

Exemplos de aplicação do encontro no meio

Soma de subconjuntos

- Considere o seguinte problema: dados N inteiros $x_1, x_2, \ldots, x_N$, e um inteiro x, determine se existem k $(1 \leq k \leq N)$ inteiros $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ tais que

$$x = x_{i_1} + x_{i_2} + \ldots + x_{i_k}$$

- Uma solução por força bruta avaliaria todos os 2^N subconjuntos possíveis, de modo que a complexidade seria igual a $O(N2^N)$, sendo viável para valores de $N \approx 20$
- Contudo, o encontro no meio pode ser utilizado para obter uma complexidade igual a $O(N2^{N/2} \log N)$, de modo que o problema poderia ser resolvido para valores de $N \approx 40$

Soma de subconjuntos

- A solução iniciaria subdividindo os inteiros em dois grupos

$$G_1 = \{x_1, x_2, \ldots, x_{\lfloor N/2 \rfloor}\}$$

e

$$G_2 = \{x_{\lfloor N/2 \rfloor + 1}, x_{\lfloor N/2 \rfloor + 2}, \ldots, x_N\}$$

- Em seguida, geraria os subconjuntos S_1 e S_2 das somas de todos os subconjuntos de G_1 e G_2, respectivamente
- Assim, ambos subconjuntos teriam, aproximadamente, $2^{N/2}$ elementos
- Em seguida, para cada elemento $s \in S_1$, uma busca binária tentaria localizar o elemento $r = x - s$ no conjunto S_2
- Assim, caso $r \in S_2$, a solução seria $x = s + r$, e a complexidade seria $O(N2^{N/2} \log N)$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0, 2\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0, 2\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0, 2, 4\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0, 2, 4, 5\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
S_1 = \{-3, -1, 0, 2, 4, 5, 7\}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
\begin{aligned}
S_1 &= \{-3, -1, 0, 2, 4, 5, 7\} \\
S_2 &= \{-1, 0, 3, 4, 7, 8, 11, 12\}
\end{aligned}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
\begin{aligned}
S_1 &= \{-3, -1, 0, 2, 4, 5, 7\} \\
S_2 &= \{-1, 0, 3, 4, 7, 8, 11, 12\}
\end{aligned}
$$

$$
\begin{aligned}
s &= -3 \\
r &= 13
\end{aligned}
$$

Visualização do algoritmo para soma de subconjuntos

$$
\begin{aligned}
xs &= \{2, 5, -3, 4, -1, 8\} \\
x &= 10
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= \{2, 5, -3\} \\
G_2 &= \{4, -1, 8\}
\end{aligned}
$$

$$
\begin{aligned}
S_1 &= \{-3, -1, 0, 2, 4, 5, 7\} \\
S_2 &= \{-1, 0, 3, 4, 7, 8, 11, 12\}
\end{aligned}
$$

$$
\begin{aligned}
s &= -1 \\
r &= 11
\end{aligned}
$$

Implementação da soma de subconjuntos

```cpp
#include <bits/stdc++.h>

using namespace std;

set<int> subset_sum(const vector<int>& xs)
{
    set<int> s;

    for (size_t i = 0; i < (1ul << xs.size()); ++i) {
        int sum = 0;

        for (size_t j = 0; j < xs.size(); ++j)
            if ((1 << j) & i)
                sum += xs[j];

       s.insert(sum);
    }

    return s;
}
```

Implementação da soma de subconjuntos

```
22 bool solve(int x, const vector<int>& xs)
23 {
24     int N = xs.size();
25
26     vector<int> g1(xs.begin(), xs.begin() + N/2);
27     vector<int> g2(xs.begin() + N/2, xs.end());
28
29     auto s1 = subset_sum(g1), s2 = subset_sum(g2);
30
31     for (auto s : s1)
32         if (s2.count(x - s))
33             return true;
34
35     return false;
36 }
```

- *4sum* é um problema comum em entrevistas de emprego
- Dado um vetor de inteiros $xs = \{x_1, x_2, \ldots, x_N\}$, determine $a, b, c, d \in xs$ tais que $a + b + c + d = 0$, se existirem
- Observe que um mesmo elemento de xs pode ser escolhido mais de uma vez
- Uma solução *naive* computaria e checaria todas as $O(N^4)$ quádruplas
- Uma solução, também de busca completa, computaria $O(N^3)$ triplas (a, b, c) e procuraria pelo elemento $d = -(a + b + c)$ em xs usando busca binária, melhorando a complexidade para $O(N^3 \log N)$
- Observado que $a + b = -(c + d)$ e usando *hashes* em conjunto com o encontro no meio, é possível resolver este problema com complexidade $O(N^2)$

Implementação de *4sum* em $O(N^2)$

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<int> _4sum(const vector<int>& xs)
{
    using ii = pair<int, int>;
    unordered_map<int, ii> ps;

    for (auto x : xs)
        for (auto y : xs)
            ps[x + y] = ii(x, y);

    for (auto [s, p] : ps)
        if (ps.count(-s))
            return { p.first, p.second, ps[-s].first, ps[-s].second };

    return { };
}
```

Referências

1. **LAARKSONEN**, Antti. *Competitive Programmer's Handbook*, 2017.
2. **HALIM**, Steve; **HALIM**, Felix. *Competitive Programming 3*, Lulu, 2013.
3. InfoArena. Coding contest trick: Meet in the middle, acesso em 23/03/2020.