Pencarian Biner Lanjutan

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Memahami konsep binary search the answer
- Memahami konsep ternary search

Mulai dari bab ini, seluruh kode program akan dituliskan dalam bahasa pseudo-C++.

Binary Search the Answer

- Pada dasarnya, teknik ini menggunakan konsep *binary search* dalam menebak suatu jawaban, kemudian membuat keputusan untuk memperkecil ruang pencarian kita dalam mencari jawaban yang optimal.
- Teknik ini sebenarnya tidak sulit untuk diaplikasikan jika kita sudah memahami teknik dasar *binary search*. Bahkan, beberapa orang ada yang dapat mengaplikasikan teknik ini secara tidak sadar sebelum mempelajarinya.

Binary Search

- Mari kita lihat kembali bagaimana konsep dasar dari *binary search* itu sendiri.
- Contoh soal: Diberikan suatu *array* A sepanjang N yang elemennya terurut dari kecil ke besar. Tentukan indeks dengan nilai terkecil pada *array* tersebut yang nilainya $\geq X$ dalam waktu $O(logN)$.

Binary Search (lanj.)

- Soal ini dapat diselesaikan menggunakan *binary search*:
  - Misalkan diberikan $N = 8$, $A = [1, 2, 3, 5, 8, 13, 21, 34]$, dan $X = 7$.
  - Mari kita definisikan B sebagai *array* sepanjang N dengan $B[i]$ bernilai:
    - 0 jika $A[i] < X$, atau
    - 1 jika $A[i] \geq X$.
  - Dengan menggunakan *array* B, kita dapat mengubah soal yang diberikan menjadi sebagai berikut:
    - Tentukan indeks pertama yang bernilai 1 pada *array* yang hanya berisikan angka 0 dan 1, serta nilainya terurut dari kecil ke besar.

Binary Search (lanj.)

- Soal sebelumnya dapat diselesaikan menggunakan kode berikut.

```
void check(int v, int X) {
  return v >= X;
}

int solve(int N, vector<int> A, int X) {
  int l = 0, r = N - 1;
  while (l < r) {
    int mid = (l + r) >> 1;
    if (check(A[i], X)) r = mid;
    else l = mid + 1;
  }
  return l;
}
```

Binary Search the Answer

- Secara tidak langsung, dalam menyelesaikan soal tersebut, kita sudah mengaplikasikan teknik ini.
- Kita melakukan *binary search* pada indeks *array* B, dengan indeks tersebut akan menjadi jawaban.
- Untuk memperkecil kemungkinan jawaban, kita memeriksa suatu tebakan dengan menggunakan fungsi `check` yang membantu kita dalam membuat keputusan apakah harus memeriksa indeks di kiri atau kanan dari tebakan pada saat tersebut.

Contoh Soal

- Terdapat N barang dengan harga yang berbeda-beda dinomori 1 sampai N.
- Kita diminta untuk membagi N barang ini menjadi $K (2 \leq K \leq N)$ kelompok, sedemikian sehingga setiap barang berada dalam tepat satu kelompok, dan setiap kelompok berisi setidaknya satu barang yang memiliki nomor yang berurutan.
- Kemudian, $K - 1$ teman kita secara satu per satu akan memilih satu kelompok yang belum dipilih sebelumnya, dan mengambil seluruh barang yang berada di dalam kelompok tersebut.
- Kita akan mengambil seluruh barang di satu kelompok yang tersisa. Karena teman kita rakus, kita akan mendapatkan kelompok dengan total harga barang yang paling kecil.
- Tentukan total harga barang maksimum yang dapat kita peroleh dengan mengatur pembagian kelompok barang seoptimal mungkin.

Solusi

- Kita dapat menggunakan fungsi f sedemikian sehingga $f(x) = 1$ jika kita dapat mendapatkan kelompok dengan total harga barang setidaknya x, atau $f(x) = 0$ jika tidak.
- Perhatikan bahwa fungsi ini terurut dari besar ke kecil.
  - Jika kita dapat mendapatkan kelompok dengan total harga barang setidaknya x, maka kita juga dapat mendapatkan kelompok dengan total harga barang setidaknya $x - 1$.
  - Sebaliknya, jika kita tidak dapat mendapatkan kelompok dengan total harga barang setidaknya x, maka kita juga tidak dapat mendapatkan kelompok dengan total harga barang setidaknya $x + 1$.

Solusi (lanj.)

- Jika kita ingin mendapatkan kelompok dengan total harga barang setidaknya x, maka kita harus membagi kelompok barang sedemikian sehingga seluruh kelompok memiliki total harga barang setidaknya x.
  - Sehingga, fungsi ini dapat diimplementasikan dengan memeriksa apakah kita dapat membagi kelompok barang yang memenuhi kondisi tersebut.
  - Kita dapat menggunakan algoritme *greedy*.
- Kompleksitas solusi ini adalah $O(N \times \log(\sum H))$ dengan $\sum H$ adalah total harga barang.

Solusi (lanj.)

```
bool f(int N, int K, const vector<int>& H, int X) {
  int current_sum = 0;
  for (int i = 0; i < N; ++i) {
    if (current_sum + H[i] >= X) {
      // Dibentuk kelompok dengan total harga barang >= X.
      --K; current_sum = 0;
    } else {
      current_sum += H[i];
    }
  }
  return (K <= 0 ? 1 : 0);
}

int solve(int N, int K, vector<int> H) {
  int l = 0, r = accumulate(H.begin(), H.end(), 0);
  while (l < r) {
    int mid = (l + r) >> 1;
    if (f(N, K, H, mid)) l = mid;
    else r = mid - 1;
  }
  return l;
}
```

Ternary Search

- Jika *array* yang dimiliki tidak terurut namun *unimodal*, kita dapat menggunakan *ternary search* untuk mencari sebuah elemen pada *array* tersebut dalam $O(\log(N))$.
- *Array* **unimodal** adalah *array* yang memiliki hanya satu puncak. Lebih resminya, *Array* A adalah *unimodal* jika terdapat suatu indeks x yang memenuhi:
  - Untuk seluruh a, b yang memenuhi $a < b \leq x$, $A[a] < A[b]$.
  - Untuk seluruh a, b yang memenuhi $x \leq a < b$, $A[a] > A[b]$.
- Teknik serupa juga dapat digunakan bila *array* memiliki hanya satu lembah.

Ternary Search (lanj.)

- Jika kita tahu bahwa puncak *array* berada di antara indeks L dan R, maka jika kita memiliki dua indeks M_1 dan M_2 ($L < M_1 < M_2 < R$)
  - Jika $A[M_1] < A[M_2]$, maka puncak *array* tidak mungkin berada di sisi kiri (antara indeks L dan M_1). Maka kita dapat fokus melakukan pencarian di antara indeks $M_1 + 1$ dan R.
  - Jika $A[M_1] > A[M_2]$, maka puncak *array* tidak mungkin berada di sisi kanan (antara indeks M_2 dan R). Maka kita dapat fokus melakukan pencarian di antara indeks L dan $M_2 - 1$.
  - Jika $A[M_1] = A[M_2]$, maka puncak *array* tidak mungkin berada di kedua sisi. Maka kita dapat fokus melakukan pencarian di antara indeks $M_1 + 1$ dan $M_2 - 1$.
- Agar ukuran rentang pencarian berkurang menjadi paling banyak $\frac{2}{3}$ kali ukuran rentang pencarian sebelumnya pada seluruh kasus, kita dapat menggunakan $M_1 = L + \frac{1}{3}(R - L)$ dan $M_2 = R - \frac{1}{3}(R - L)$.