[B8] 二分探索とハッシュ

大崎 博之

関西学院大学 理工学部 情報科学科

ohsaki@kwansei.ac.jp

[B8] 二分探索とハッシュ

- 1. **線形探索** (linear search)
- 2. **二分探索** (binary search)
- 3. **ハッシュ法** (hash)

これらを学ぶことに何の意味があるのか?

探索問題 (search problem)

- **入力**: **n 個の値の順列** $(a_1, a_2, \ldots, a_n)$**、探索する値** v
- **出力**: **順列のインデックス** i **ただし** $a_i = v$

線形探索 (linear search)

時間計算量: $O(n)$

```
LINEAR-SEARCH(A, v)
1  for i ← 0 to length[A] - 1
2       do if A[i] = v
3             then return i
4  return ϕ
```

1. **先頭から順番に探索し、見つかればそこで終了**

二分探索 (binary search) (再帰による実現)

時間計算量: $O(logn)$

値の昇順に順列がソートされていることが前提

```
BINARY-SEARCH(A, v, low, high)
1  if high < low
2     then return ϕ
3  mid ← (low + high) / 2
4  if A[mid] > v
5     then return BINARY-SEARCH(A, v, low, mid - 1)
6  if A[mid] < v
7     then return BINARY-SEARCH(A, v, mid + 1, high)
8  return mid
```

1. **中央の値が v に一致すればそこで終了**
2. **中央の値が v より大きければ左半分を二分探索**
3. **中央の値が v より小さければ右半分を二分探索**

二分探索 (binary search) (繰り返しによる実現)

時間計算量: $O(logn)$

```
BINARY-SEARCH(A, v)
 1  low ← 0
 2  high ← length[A] - 1
 3  while low <= high
 4  do mid ← (low + high) / 2
 5     if A[mid] = v
 6        then return mid
 7     if A[mid] > v
 8        then high ← mid - 1
 9     if A[mid] < v
10        then low ← mid + 1
11  return ϕ
```

1. **中央の値が v に一致すればそこで終了**
2. **中央の値が v より大きければ左半分を二分探索**
3. **中央の値が v より小さければ右半分を二分探索**

ハッシュ法 (hash) (ハッシュ表への挿入)

時間計算量: $O(1)$

```
HASH-INSERT(H, key, val)
1  i ← hash(key, length[H])
2  # スロットに空きがない時のエラーチェックが必要
3  while H[i] is occupied
4       do i ← i+1 modulus length[H]
5  H[i].KEY ← key
6  H[i].VAL ← val
```

1. **key のハッシュ値 i を計算する**
2. **i 番目のスロットから空きスロットを探索**
3. **見つかった空きスロットに key, val を保存**

ハッシュ法 (hash) (ハッシュ表を用いた検索)

時間計算量: $O(1)$

```
HASH-SEARCH(H, key)
1  i ← hash(key, length[H])
2  # スロットが一杯で key が存在しない時のエラーチェックが必要
3  while H[i] is occupied and H[i].KEY != key
4       do i ← i + 1 modulus length[H]
5  if H[i].KEY = key
6     then return H[i].VAL
7     else return ϕ
```

1. **key のハッシュ値 i を計算する**
2. **i 番目のスロットから key が見つかるまで探索**

探索を学ぶ意義

- プログラミングの本質は アルゴリズム と データ構造
  - プログラミングの成功/失敗は……
    - 良い アルゴリズム を考えられるか?
    - 良い データ構造 を考えられるか?
  - プログラミングテクニックは実は瑣末 (さまつ) な問題
  - だから大学の講義はとても重要!

探索 = 最も重要なアルゴリズムの一つ

→ 設計した アルゴリズム を実現するための技術が必要

(実用上は、探索のプログラムを自分で書く必要はない)