Fast campus

컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 42강 - 인덱스 트리

학습 목표

인덱스 트리

1) 인덱스 트리의 필요성과 구현 방법에 대해서 이해합니다.
2) 인덱스 트리를 활용하여 구간 합을 구하는 방법을 알 수 있습니다.

인덱스 트리

트리 구조로 구간 합 구하기

1) 세그먼트 트리는 구현하는 과정이 복잡하고 어렵다는 점에서 구간 합을 더 쉽게 구할 방법이 필요합니다.
2) 인덱스 트리는 구현이 매우 간단하다는 특징이 있습니다.
3) 인덱스 트리를 활용하여 구간 합을 구하는 과정도 $O(logN)$의 시간 복잡도를 가집니다.
4) 인덱스 트리는 세그먼트 트리에 비해서 메모리 효율성이 높습니다.

인덱스 트리

특정한 숫자의 마지막 비트 구하기

특정한 숫자 A의 가장 마지막 비트를 구하고 자 할 때는 A & -A를 구하면 됩니다.

EX) A가 14일 때 이를 비트 형태로 표현하면 다음과 같습니다.

A = 00000000 00000000 00000000 00001110
-A = 11111111 11111111 11111111 11110010

A & -A = 00000000 00000000 00000000 00000010

인덱스 트리

마지막 비트를 이용해 트리 구조 만들기

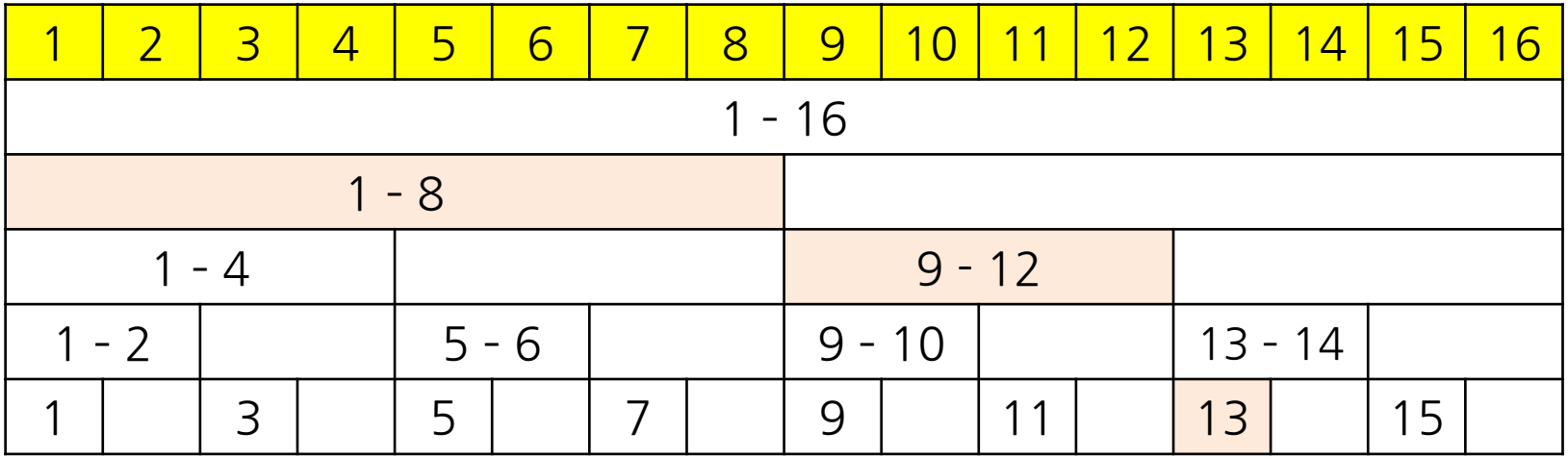
각 인덱스에 대하여 마지막 비트를 '내가 저장하고 있는 값들의 개수'로 계산합니다.

<table>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td></tr>
<tr><td colspan="16">1 - 16</td></tr>
<tr><td colspan="8">1 - 8</td><td colspan="8"></td></tr>
<tr><td colspan="4">1 - 4</td><td colspan="4"></td><td colspan="4">9 - 12</td><td colspan="4"></td></tr>
<tr><td colspan="2">1 - 2</td><td colspan="2"></td><td colspan="2">5 - 6</td><td colspan="2"></td><td colspan="2">9 - 10</td><td colspan="2"></td><td colspan="2">13 - 14</td><td colspan="2"></td></tr>
<tr><td>1</td><td></td><td>3</td><td></td><td>5</td><td></td><td>7</td><td></td><td>9</td><td></td><td>11</td><td></td><td>13</td><td></td><td>15</td><td></td></tr>
</table>

인덱스 트리

인덱스 트리: 1부터 N까지의 구간 합 구하기 [인덱스 1부터 13까지]

마지막 비트만큼 구간을 앞으로 이동하며 합을 구하면 됩니다.



인덱스 트리

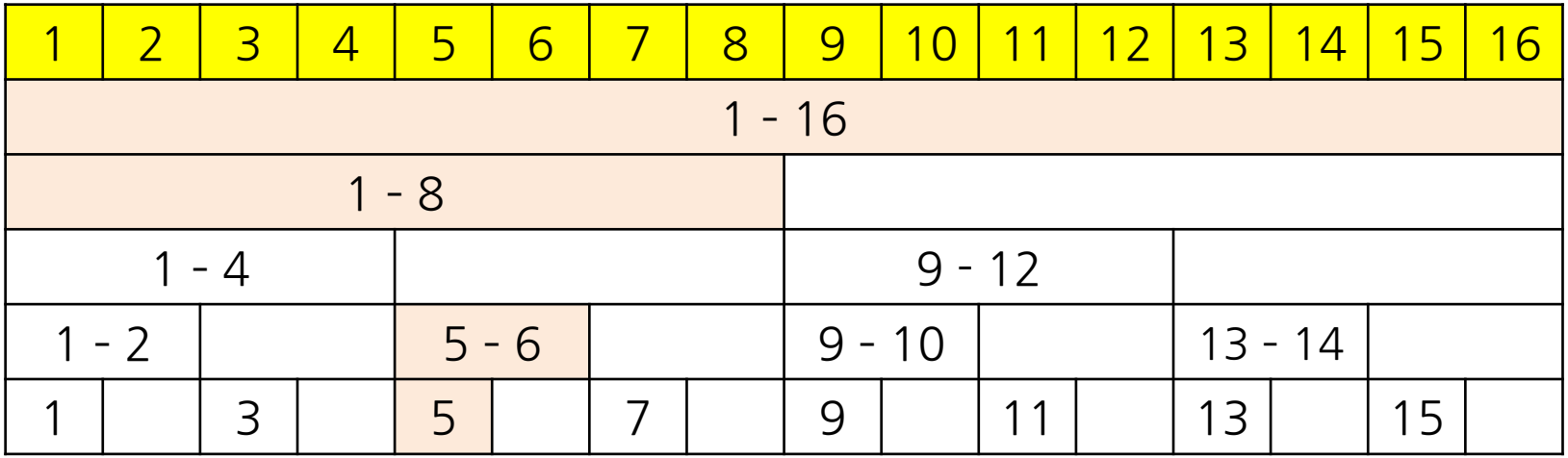
인덱스 트리: 1부터 N까지의 구간 합 구하기

```c
#include <stdio.h>
#define NUMBER 7

int tree[NUMBER];

int sum(int i) {
  int res = 0;
  while (i > 0) {
    res += tree[i];
    // 마지막 비트만큼 빼면서 앞으로 이동
    i -= (i & -i);
  }
  return res;
}
```

인덱스 트리

인덱스 트리: 특정 인덱스 수정하기 [인덱스 5]

마지막 비트만큼 구간을 뒤로 이동하며 값을 수정하면 됩니다.



Fast campus

인덱스 트리

인덱스 트리: 특정 인덱스 수정하기

```
void update(int i, int dif) {
  while (i <= NUMBER) {
    tree[i] += dif;
    // 마지막 비트만큼 더하면서 뒤로 이동
    i += (i & -i);
  }
}
```

Fast campus

인덱스 트리

인덱스 트리: 구간 합 계산 함수 구현하기

```
int getSection(int start, int end) {
  return sum(end) - sum(start - 1);
}
```

인덱스 트리

인덱스 트리 사용해보기

```
int main(void) {
  update(1, 7);
  update(2, 1);
  update(3, 9);
  update(4, 5);
  update(5, 6);
  update(6, 4);
  update(7, 1);
  printf("1부터 7까지의 구간 합: %d\n", getSection(1, 7));
  printf("인덱스 6의 원소를 +3만큼 수정\n");
  update(6, 3);
  printf("4부터 7까지의 구간 합: %d\n", getSection(4, 7));
  system("pause");
}
```

배운 내용 정리하기

인덱스 트리

1) 인덱스 트리에서의 구간 합 계산 및 원소 수정 과정은 $O(logN)$의 시간 복잡도를 가집니다.