- 이진탐색 –

: 정렬된 리스트에서 검색 범위를 줄여 나가면서 검색 값을 찾는 알로리즘.

장점: 속도가 빨라 검색 반복 시 검색 범위가 절반으로 줄어든다.

단점: 정렬된 리스트에서만 사용이 가능하다.

- 동작원리: 중간값을 이용한 타깃 찾기

1) 배열의 중간 값을 가져옴.

2) 중간 값과 검색 값을 비교함.

a. 중간 값이 검색 값과 같다면 종료.

b. 중간 값 < 검색 값, 중간값 기준 배열의 우측 구간을 대상으로 탐색.

c. 중간 값 > 검색 값, 중간값 기준 배열의 좌측 구간을 대상으로 탐색

3) 값을 찾거나 간격이 비어있을 때까지 반복.

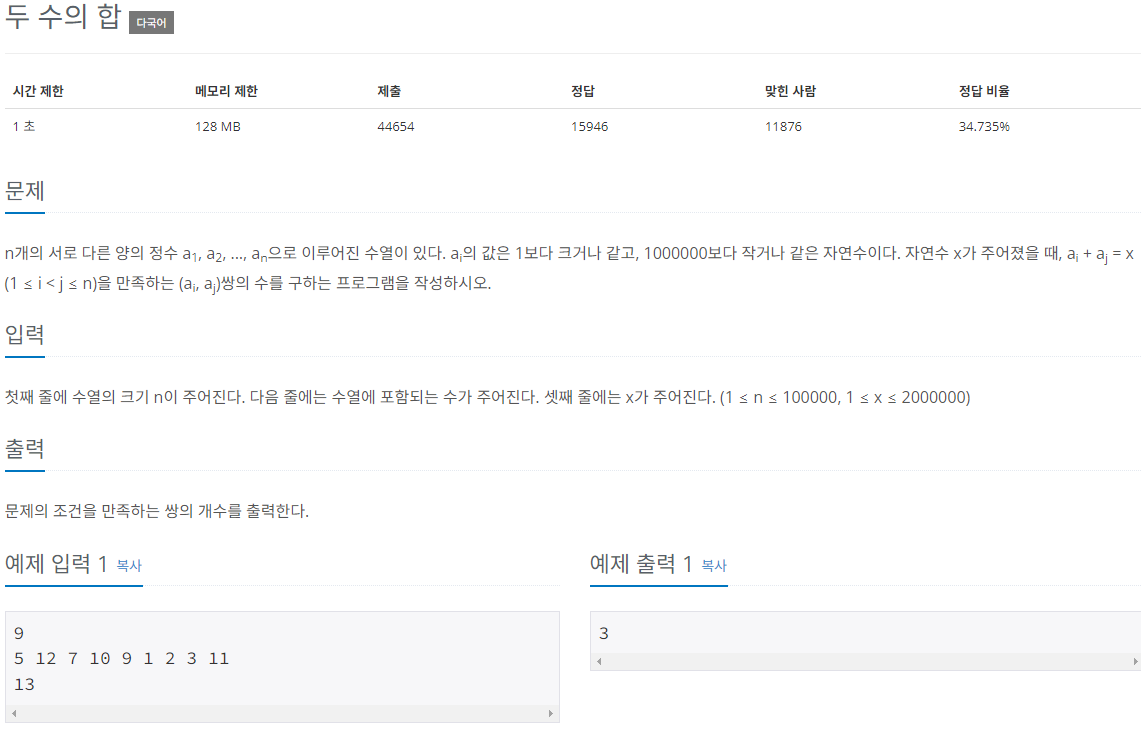
- 중간값 구하는 계산식

mid = (start + end)//2

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

<https://velog.io/@madfinger/BinarySearch%EC%9D%B4%EC%A7%84-%ED%83%90%EC%83%89-%ED%8C%8C%EC%9D%B4%EC%8D%AC>

백준 3273번 두 수의 합 문제

from sys import stdin

from bisect import bisect\_right, bisect\_left # 이진 분할 알고리즘을 모듈사용

n = int(stdin.readline()) # 전체 수열 크기

arr = list(map(int, stdin.readline().split())) # 오름차순 전체 수열의 수

arr.sort()

x = int(stdin.readline()) 두 쌍의 합이 되는 자연수

cnt = 0

for i in range(n):

l, r = bisect\_left(arr, x - arr[i]), bisect\_right(arr, x - arr[i])

if l < n:

if arr[i] != arr[l] and l != r:

cnt += 1

print(cnt *// 2)*

- bisect 모듈 설명: https://www.folivoralab.com/83

- 투 포인터 -

import sys

n = int(input()) # 수열의 크기

numbers = sorted(list(map(int, sys.stdin.readline().split()))) # 전체 수열의 수(오름차순 정렬)

x = int(input()) # 두 쌍의 합이 되는 자연수

answer = 0 # 0부터 시작하는 초기화된 두 쌍을 카운트한 수

left, right = 0, n-1 # 왼쪽(수열의 start인 0으로 초기화), 오른쪽 변수(수열의 end인 끝으로 초기화 )

# left, right를 인덱스로 넣어주어 두 변수가 서로 교차할 때까지 두 쌍의 합이 x인 것을 count함

while left < right :

temp = numbers[left] + numbers[right] # 특정한 합을 가지는 수열의 쌍을 찾기위한 변수 설정

if temp == x : # 두 쌍의 합인 자연수 x가 맞을 경우

answer += 1 # 쌍이 맞으므로 카운트 +1

left += 1 # x가 맞으므로 무시하고 다음 값으로 넘어감

elif temp < x : #

left += 1 # 두 합이 x 보다 작으므로 더 큰 값이 필요.

else: # 그 외 조건이기 때문에 temp > x 임

right -= 1 # 두 합이 x 보다 크므로 더 작은 값이 필요.

print(answer) # 두 쌍의 합을 카운트한 값 출력

- DP 동적 계획법: 다이나믹 프로그래밍 –

: 하나의 큰 문제를 작은 문제로 나누고, 그 작은 문제를 해결하여 큰 문제의 답을 도출해내는 기법. 또는 작은 문제를 해결하는 과정에서 중복되는 연산을 수행하지 않는 기법.

대표적 예시 피보나치 수

- Fn = Fn-1 + Fn-2 -> Fn 큰 문제, Fn-1과 Fn-2 작은 문제

문제풀이 방식

1) Top - Down : 재귀호출 방식

- 큰 문제를 작은 문제로 나눈다. -> F(n-1), F(n-2)

- 작은 문제를 푼다. -> F(n-1) + F(n-2)

2) Bottom – UP

- 문제를 크기가 작은 문제부터 차례대로 쓴다.

- 문제의 크기를 조금씩 크게 만들면서 문제를 푼다.

- 작은 문제를 풀면서 큰 문제의 답을 구한다.

출처: <https://developer-mac.tistory.com/77>

[백준 1149번 RGB거리]



import sys

n = int(sys.stdin.readline())

rgb = []

for \_ in range(n):

rgb.append(list(map(int,sys.stdin.readline().rstrip().split())))

for i in range(1,n):

rgb[i][0] = min(rgb[i-1][1],rgb[i-1][2]) + rgb[i][0]

rgb[i][1] = min(rgb[i - 1][0], rgb[i - 1][2]) + rgb[i][1]

rgb[i][2] = min(rgb[i - 1][0], rgb[i - 1][1]) + rgb[i][2]

print(min(rgb[n-1]))

- 빨간색일 때, 현재 집의 비용에 이전 순서의 초록&파랑색 집의 비용중 최소 값을 더함.

- 초록색일 경우, 현재 집의 비용에 이전 순서의 빨강&파랑색 집의 최소 비용을 더함

- 파랑색일 경우, 현재 집의 비용에 이전 순서의 빨강&초록색 집의 최소 비용을 더함

- 마지막 집의 비용이 담긴 배열에는 마지막 집의 색상 별로 전체 비용이 담김

출처: https://programmers-story.tistory.com/entry/%EB%B0%B1%EC%A4%80%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%981149%EB%B2%88%ED%8C%8C%EC%9D%B4%EC%8D%AC-RGB%EA%B1%B0%EB%A6%AC