

자료구조 - 201921725 안성현

버스 탑승 스케줄링

<2021/04/05>

문제 소개 및 접근법

1-1] 버스 문제

① 문제

사람들에게 서울역에서 목장까지 이동하는 버스편을 제공하기로 한다. 사람들은 총 n 명, 버스는 모두 m 대, 그리고 버스 한 대에는 최대 c 명까지 탑승할 수 있다.

각 버스는 최대 c 명까지 탑승한 후 목장으로 출발한다. 버스에 탑승한 참가자는 버스가 출발할 때까지 오랜 시간을 기다려야 할 수도 있다.

이 때, 참가자들이 버스에 탑승한 후 버스가 출발할 때까지 기다리는 시간의 최대값을 최소화하라. (단 $n \leq mc$, $c \leq n$ 은 보장한다)

ex) 6 3 2 / 1 1 10 14 4 3이 입력이면 4가 출력이 된다. 버스A[1,1] 버스B[3,4] 버스C[10,14]로 탑승 시켰을 때보다 적은 최대 대기 시간(최대값)은 존재하지 않는다.

② 접근법

▶ 사람들이 서울역에 오는 시각은 정해져있다. 버스 관계자는 m 개의 버스를 가지고 모든 사람들이 최대한 적게 대기하도록 만들고 싶다. 그래서 '최대 대기 시간'을 임의로 정하고 그 시간 안에 사람들이 버스를 나눠 탈 수 있는지 조사한다. 여기서 '최대 대기 시간'은 (마지막에 오는 사람의 시각)-(처음 오는 사람의 시각)으로, 0~(가장 늦은 시각) 사이에 존재한다. 만약 만족한다면 '최대 대기 시간'을 낮춰서 다시 나눠 탈 수 있는지 조사한다. 이 작업을 반복해서 더이상 사람들이 나눠 탈 수 없을 때까지 조사한다. 그럼 나눠 탈 수 없기 바로 전의 '최대 대기 시간'이 최적의 대기 시간이 될 것이다.

'최대 대기 시간' 개념을 사용하니까 가정이 필요하다. '최대 대기 시간'을 초과해서 사람이 오면 앞 버스는 가버린다는 점이다. 즉 m 대의 버스로 사람들을 모두 탑승시키에는 충분하지만, 누군가 너무 늦게 온다면 m 대를 다 놓쳐버릴 수 있다. 위 사항을 고려해서 코드를 작

성해야 한다.

ex) 사람2명,버스1대,좌석2개일 때 최대 대기 시간이 5이고 사람들이 (10,20)에 각각 버스 타러 온다. -> 10분기다릴 수 없으니 사람(10)만 버스타고 출발한다.

위의 경우, 2대의 버스가 와야지 최대 대기 시간 5를 만족한다. -> 1대의 버스로 처리하기 위해서는 최대 대기 시간을 늘려서 처리해야 될 것이다. -> 10이 최적의 대기시간이 됨

소스 코드와 알고리즘 설명

2-1] 서울역에 온 사람들의 시각 정렬

① 코드

```
void swap_elements(int* list, int index1, int index2) {
    int temp = list[index1];
    list[index1] = list[index2];
    list[index2] = temp;
}

int partition(int* list, int start, int end) {
    int i, b;
    i = b = start;
    int p = end;

    while (i < p) {
        if (list[i] <= list[p]) {
            swap_elements(list, i, b);
            b += 1;
        }
        i += 1;
    }
    swap_elements(list, b, p);
    p = b;

    return p;
}

int quick_sort(int* list, int start, int end) {
    if (end - start < 1)
        return 0;

    int p = partition(list, start, end);
    quick_sort(list, start, p - 1);
    quick_sort(list, p + 1, end);
}
```

② 알고리즘 설명

▶ 대기 시간은 (늦게 온 사람-빨리 온 사람)이므로 음수가 발생하면 안 된다. 따라서 정렬이 필요하다.

정렬하지 않으면 빨리 온 사람과 늦게 온 사람을 쉽게 구분하기가 힘들다.

또한 이후에 사용할 탐색 알고리즘의 성능에도 영향을 끼친다.

필자는 평균 시간복잡도가 $O(n \log n)$ 을 만족하는 '퀵 정렬'을 사용하였다.

2-2] 첫 번째 '최대 대기 시간' 구하기

① 코드

```
if ((n % c) == 0) { // n=m*c, n<m*c (n = k*c)
    if ((m * c - n) >= c) // ----- (m-α) buses can go
        M = n / c;
    for (i = 0; i < M; i++) {
        tmp = time_list[idx + (c - 1)] - time_list[idx];
        idx += c;
        //find max
        if (tmp >= fmd)
            fmd = tmp;
    }
}
else { // n<m*c (n != k*c)
    if ((m * c - n) >= c) // ----- (m-α) buses can go
        M = (n / c) + 1;
    for (i = 0; i < M; i++) {
        if (i == M - 1) {
            tmp = time_list[idx + ((n % c) - 1)] - time_list[idx];
        }
        else {
            tmp = time_list[idx + (c - 1)] - time_list[idx];
            idx += c;
        }
        //find max
        if (tmp >= fmd)
            fmd = tmp;
    }
}
```

② 알고리즘 설명

▶ 최대 대기 시간을 처음 정할 때는 버스를 최대한 낭비하지 않고 사람들을 태웠을 때를 기준으로 한다.

[사람들을 버스에 정원으로 채워도 남는 자리 \geq 버스의 좌석 자리] $[(m * c - n) \geq c]$ 는 버스가 낭비되는 경우이다. 이럴 경우 m 을 일시적으로 변경한다.

ex) 8 / 2 / 4 이고 [1,2,3,4,5,6,7,8]이면 무조건 [1234/5678]로 태움 -> 최대 대기 시간 3

ex) 8 /3 /4 이고 [1,2,3,4,5,6,7,8]이면 무조건 [1234/5678]로 태움 -> 최대 대기 시간 3
<나머지 한 버스는 무시하기 위해 8/2/4로 바뀌서 풀었음>

ex) 8 /3 /3 이고 [1,2,3,4,5,6,7,8]이면 무조건 [123/456/78]로 태움 -> 최대 대기 시간 2

ex) 8 /5 /3 이고 [1,2,3,4,5,6,7,8]이면 무조건 [123/456/78]로 태움 -> 최대 대기 시간 2
<나머지 두 버스는 무시하기 위해 8/3/3으로 바뀌서 풀었음>

위 예시에 의하면 2를 '첫 번째 최대 대기 시간'으로 정하게 된다. 이제 (0~2)의 범위에서 최적의 '최대 대기 시간'을 찾으면 된다. 이렇게 미리 정해놓지 않으면 '최대 대기 시간'을 만족하는지 비교하는 횟수가 훨씬 늘어날 것이다.

2-3] 최적의 '최대 대기 시간' 탐색하기

① 코드

```
int binary_search_compare(int* list, int first, int last, int fmd) {  
  
    if ((last == 0) || (first == fmd)) // 처음 값 혹은 마지막 값 처리 (최종의 경우),  
조건문에서 생략했지만 first=last임  
        return first; // 최종의 경우까지 오면 나올 수밖에 없음  
  
    int mid = (first + last) / 2;  
  
    int bus_mid = bus_num(list, mid); // 최대 대기시간(max_diff)이 mid일 때, bus값  
  
    if (mid == 0) { // first:0, last:1이어서 mid가 0이면 'Main Part'의 연산이 불가능  
-> 따로 처리  
        if (bus_mid <= m) // max_diff가 0일 때 조건 만족 -> 0이 최적의 최대  
대기시간  
            return 0;  
        else  
            return 1;  
    }  
  
    // Main Part  
    if (bus_mid <= m) { // max_diff가 0이 아닌 mid일 때, bus<=m [현재 조건  
만족(적거나 같음)]  
        if (bus_num(list, mid - 1) > m) // max_diff가 mid일 때, bus>m [바로 전일  
때 조건 불만족]  
            return mid; // best_time은 mid  
  
        else // [바로 전일 때 조건 만족]  
            binary_search_compare(list, first, mid - 1, fmd); // '이전 값들  
중 하나'에서 재조사  
    }  
    else // max_diff가 0이 아닌 mid일 때, bus>m [현재 조건 불만족]  
        binary_search_compare(list, mid + 1, last, fmd); // '다음 값들 중  
하나'에서 재조사  
}
```

② 알고리즘 설명

▶ 어떠한 것이 최적의 '최대 대기 시간'인지 알아보기 위해 간단한 로직을 알아 보겠다.

범위 [0 1 2 3 4 5 6 7 8 9 10] 중 [최대 대기 시간:0]일 때 만족한다면 1은 조사할 필요도 없이 만족한다.

ex) 사람 두 명이 같은 시각에 도착해서 정원이 두 명인 버스를 탔다. 사실상 그냥 출발해도 상관이 없다. 즉 (최대)대기 시간이 0일때도 아무 상관이 없다. 그런데 기사님이 1분뒤에 출발한다고 한다. (최대)대기 시간이 1이 된 것이다. 그냥 출발하나 1~10분을 기다리고 출발하나 이 사람들은 항상 버스를 탑승하는 조건에 만족한다.

즉 k일 때 만족한다면 k+1일 때도 만족한다.

이를 바꿔서 생각하면 p일 때 만족하고 p-1일 때 만족하지 않는다면 p가 '최적의 최대 대기 시간'이 된다.

ex) [0:불만족, 1:불만족, 2:불만족, 3:불만족, 4:만족, 5:만족,]라면 4가 최적의 '최대 대기 시간'

최적의 '최대 대기 시간'은 (0~10)에서 찾고 3이 정답이라고 가정해보겠다. 만약 10에서 1씩 줄어서 만족하는지 찾는다면 8번을 찾게 된다. 하지만 이분 탐색을 이용하면 3번만에 찾을 수 있다. 실제로 순차 탐색과 이분 탐색의 시간 복잡도를 비교하면 $O(n)$ 과 $O(\log n)$ 의 차이가 발생한다. 만일 범위가 (0~1,000,000,000)이었다면 999,999,998번과 29번의 차이라는 의미이다. 따라서 필자는 이분 탐색 알고리즘을 응용해서 문제를 풀었다.

0. 이분 탐색 알고리즘 중 mid를 최대 대기 시간으로 임의 설정한다.

1. mid가 최대 대기 시간일 때, 조건을 만족하면 mid-1이 최대 대기 시간일 때도 만족하는지 조사한다.

둘 다 만족한다면 mid는 최적의 최대 대기 시간이 아니다.

이 때는 last를 (mid-1)로 만들어서 mid보다 적은 범위의 수를 살펴본다.

<'불만족-만족'의 경우는 앞쪽에 위치하므로>

ex) [0:?, 1:?, 2:?, 3:?, 4:만족, 5:만족,]일 때 mid가 5일 경우

2. mid가 최대 대기 시간일 때, 조건을 만족하고 mid-1이 최대 대기 시간일 때는 불만족한다면 mid를 반환한다. <정답 찾음>

ex) [0:?, 1:?, 2:?, 3:불만족, 4:만족, 5:?,]일 때 mid가 4일 경우

3. mid가 최대 대기 시간일 때, 조건을 불만족한다면 first를 (mid+1)로 만들어서 mid보다 큰 범위의 수를 살펴본다.

<일단 mid가 만족하는 경우를 찾아야 함, '만족'은 항상 '불만족' 뒤에 있음>

ex) [0:?, 1:?, 2:?, 3:?, 4:?, 5:?, 6:불만족]일 때 mid가 6일 경우

예외 처리) mid가 0이 오면 mid-1이 -1이 돼서 stack over flow가 발생한다. 이때는 0일 때 조건을 만족하면 0리턴, 아니면 1을 반환한다.

<first가 0, last가 1이므로 둘 중 하나가 답>

종료 조건) last=0일 때, first=fmd(첫 번째 최대 대기 시간)일 때는 최종의 경우이다. 최종의 경우까지 온다면 답이 0 or fmd일 수밖에 없다.

ex) [0:만족, 1:만족, 2:만족, 3:만족, 4:만족, 5:만족,]일 때 last가 0일 경우

ex) [0:불만족, 1:불만족, 2:불만족, ..., 9:불만족, 10:만족]일 때 first가 10일 경우

2-4] 조건 만족 확인하기

① 코드

```
int bus_num(int* list, int max_diff) {
    int a = 0; // list[a]~list[b]는 현재 버스에 탈 수 있을지 조사할 때 필요한
              // 사람들(시각), list[a]~list[b-1]은 현재 버스에 탄 사람들
    int b = 1;
    int bus = 1;

    while (b < n) { // list[n-1]까지 존재함
        if (list[b] - list[a] <= max_diff) { // 대기시간이 최대 대기시간
            이하이다.
                if ((b - a + 1) > c) { // 사람 수가 좌석 수에 불만족
                    bus++; // 버스 늘림
                    a = b; // 다음 버스 처리 시작
                }
            }
        else { // 대기시간이 최대 대기시간 초과이다.
            bus++; // 버스 늘림
            if (bus > m) // 최대 버스 수 초과
                break; // 버스 조사 끝냄
            a = b; // 다음 버스 처리 시작
        }
        b++; // 사람 태움, 다음 경우 조사를 위해 b를 1증가
    }
    return bus;
}
```

② 알고리즘 설명

▶ 이제 조건을 만족하다는 것이 무슨 의미인지 알아 보겠다.

max_diff(최대 대기 시간)가 1이고 사람들이 서울역에 오는 시각은 [1,2,2,6]이고 버스의 수와 정원은 (3,2)라고 가정하겠다.

(1,2)는 1차이가 나니까 같은 버스에 탑승시킬 수 있다. 고로 (1,2)는 버스1에 탑승시킨다. 그리고 다음 사람도 2에 도착한다.

그러면 (1,2)는 다시 1차이가 나지만 정원이 2명이니까 사람(2)는 다음 버스(버스2)에 탑승시킨다.

마지막 사람은 6에 도착하는데 (2,6)은 4차이가 나니까 max_diff를 만족하지 못한다. 따라서 사람(6)을 다음 버스(버스3)에 탑승시킨다.

결론은 [1,2/2/6]으로 사람들을 세 버스에 탑승시키는 것에 성공했다. 이런 경우 조건을 만족한다고 말한다.

조건을 불만족하는 경우도 알아 보겠다.

max_diff(최대 대기 시간)가 3이고 사람들이 서울역에 오는 시각은 [1,7,11]이고 버스의 수와 정원은 (2,2)라고 가정하겠다.

(1,7)은 6차이가 나니까 max_diff를 만족하지 못한다. 따라서 사람(1)은 버스1에, 사람

(7)은 다음 버스(버스2)에 탑승시킨다.

(7,11)은 4차이가 나니까 역시 max_diff를 만족하지 못한다. 따라서 사람(11)은 다음 버스(버스3)에 탑승시킨다.

결론은 [1/7/11]으로 사람들을 두 버스에 탑승시키는 것에 실패했다. 이런 경우 조건을 불만족한다고 말한다.

이러한 사례들을 의사코드로 구성하면 다음과 같다.

```
while (모든 인원 조사)
  if (최대 대기시간 이하)
    if (탑승 시 좌석 수 초과)
      버스 늘리고 다음 버스에 탑승 시킴
    else
      현재 버스에 탑승 시킴
  else
    버스 늘리고 다음 버스에 탑승 시킴
    if (늘렸더니 버스 초과했다고 함)
      < 조건 불만족 > 더이상 조사할 필요가 x
  다음 사람 조사
```

// n<=mc가 보장되기 때문에, if-if문에서 버스 늘려서 초과할 사건은 발생하지 않는다.
// ex) 3 1 2, [1,3,4] max_diff가 3이면 [1,3/4]형태가 된다. 이럴 경우 버스를 2대써서 불만족하는 것처럼 보이지만 애초에 n<=mc가 성립하지 않았다.

// n<=mc가 보장되어도 else문에서 버스를 늘려서 초과할 사건은 발생한다.
// ex) 3 1 3, [1,3,5] max_diff가 3이면 [1,3/5] 형태가 된다. 마지막 사람이 5가 아니라 4였다면 만족했겠지만 너무 늦게 와서 m대를 모두 놓쳐버린 경우이다.

실제 코드는 위 알고리즘을 기반으로 구성하였다.이분 탐색 알고리즘에서 위 알고리즘을 호출하다 보니 bus를 반환해서 <만족/불만족>을 판정한다.

실행 결과와 소요 시간

3-1] 실행 결과와 소요 시간

① DEV-C++ 컴파일러 이용 실행 결과 1

```
C:\Users\tjdgdu\OneDrive\바탕 화면\buss2.exe
n,m,c입력:8 4 3
n개의 시각 입력:1 2 3 4 5 6 7 8
정렬 결과:1 2 3 4 5 6 7 8
M:3
0번째 시행: [3, 1] -> 2
1번째 시행: [6, 4] -> 2
2번째 시행: [8, 7] -> 1
first_max_diff=2
최적의 최대 대기시간: 1
(1 2) 2명 / 대기시간:1
(3 4) 2명 / 대기시간:1
(5 6) 2명 / 대기시간:1
(7 8) 2명 / 대기시간:1
n:8명, 각 버스에 탄 사람들을 합친 수:8명, m:4대, 이용 버스:4대
최대 대기 시간은 1
0.002초입니다.
-----
Process exited after 5.916 seconds with return value 18
계속하려면 아무 키나 누르십시오 . . .
```

0.002초 (정렬~'최대 대기시간=1' 출력까지)

② DEV-C++ 컴파일러 이용 실행 결과 2

```
C:\Users\tjdgdu\OneDrive\바탕 화면\buss2.exe
n,m,c입력:500 100 8
n개의 시각 입력:41 68 36 3 73 29 84 65 70 73 15 56 93 40 75 106 111 59 45 55 111 25 24 76 116 107 47 43 46 124 77 57 103
71 109 47 109 136 73 133 43 52 64 76 117 109 87 58 101 117 97 95 114 110 91 114 79 98 87 137 76 96 152 105 152 71 106 1
09 132 117 116 76 162 102 144 125 82 78 171 127 109 104 166 137 140 125 152 163 119 97 134 130 118 116 131 133 114 179 1
27 140 133 116 141 161 108 135 183 113 181 195 131 156 136 185 184 144 193 190 215 131 206 211 183 159 179 192 181 201 1
59 181 180 181 173 157 200 165 143 228 145 176 197 228 195 226 189 154 155 205 169 237 172 197 158 183 167 223 156 248 2
20 214 170 220 186 200 212 248 261 208 170 219 261 207 246 193 270 196 224 276 246 263 261 215 235 282 202 223 186 275 2
15 256 218 284 240 276 201 216 206 214 211 213 209 217 237 254 204 254 225 263 306 212 234 219 256 222 303 217 311 302 3
11 262 243 308 236 226 272 225 264 245 308 325 328 312 321 331 243 292 308 259 276 331 278 320 332 300 302 336 261 335 3
04 260 252 285 324 308 282 301 318 343 333 292 329 303 306 279 345 363 288 318 289 368 327 347 364 362 349 287 276 287 2
81 348 341 369 283 372 339 308 288 372 370 374 378 317 309 350 326 327 365 351 319 388 376 330 370 395 329 360 340 356 3
49 321 355 371 330 366 353 338 395 413 405 348 389 358 388 372 417 420 423 343 414 334 418 413 387 367 348 336 352 408 4
24 402 353 396 427 396 356 353 378 392 361 405 371 409 398 435 435 398 452 379 387 420 395 411 435 429 408 424 458 406 4
21 393 424 381 417 422 460 388 450 399 446 397 467 386 440 474 386 410 463 401 402 413 424 465 464 452 427 465 483 494 4
16 476 473 471 465 471 496 490 408 487 421 436 412 510 439 438 457 439 439 489 479 500 423 453 427 516 449 456 518 489 4
50 515 494 431 519 493 447 509 506 507 473 472 451 501 538 510 529 495 486 541 543 473 469 476 528 547 512 457 527 523 5
36 552 511 545 480 527 483 517 466 554 528 495 480 528 542 488 550 502 519 535 542 488 562 567 547 570 561 559 511 514 5
17 517 513 511 494 555 517 591 533 558 593
```

정렬 결과:3 15 24 25 29 36 40 41 43 43 45 46 47 47 52 55 56 57 58 59 64 65 68 70 71 71 73 73 73 75 76 76 76 76 77 78 79
82 84 87 87 91 93 95 96 97 97 98 101 102 103 103 103 104 105 106 106 107 108 109 109 109 110 111 111 113 114 114 114 116
116 116 116 117 117 117 118 119 124 125 125 127 127 130 131 131 131 132 133 133 133 134 135 136 136 137 137 140 140 141
143 144 144 145 152 152 152 154 155 156 156 157 158 159 159 161 162 163 165 166 167 169 170 170 171 172 173 176 179 179
180 181 181 181 181 183 183 183 184 185 186 186 189 190 192 193 193 195 195 196 197 197 200 200 201 201 202 204 205 206
206 207 208 209 211 211 212 212 213 214 214 215 215 215 215 216 217 217 218 219 219 220 220 222 223 223 224 225 225 226 226
228 228 234 235 236 237 237 240 243 243 245 246 246 248 248 252 254 254 256 256 259 260 261 261 261 261 262 263 263 270
272 275 276 276 276 276 278 279 281 282 282 283 284 284 285 287 287 288 288 289 292 292 300 301 302 302 303 303 304 306
306 308 308 308 308 308 308 311 311 312 317 318 318 319 320 321 321 324 325 326 327 327 328 329 329 330 330 331 331 332
333 334 335 336 336 338 339 340 341 343 343 345 347 348 348 348 349 349 350 351 352 353 353 353 355 356 356 358 360 361

```

C:\Users#tjdgu#OneDrive#바탕 화면#buss2.exe
34번째 시행: [332, 328] -> 4
35번째 시행: [340, 333] -> 7
36번째 시행: [348, 341] -> 7
37번째 시행: [353, 349] -> 4
38번째 시행: [363, 355] -> 8
39번째 시행: [370, 364] -> 6
40번째 시행: [376, 370] -> 6
41번째 시행: [387, 378] -> 9
42번째 시행: [395, 388] -> 7
43번째 시행: [402, 396] -> 6
44번째 시행: [409, 402] -> 7
45번째 시행: [416, 410] -> 6
46번째 시행: [422, 417] -> 5
47번째 시행: [427, 423] -> 4
48번째 시행: [438, 427] -> 11
49번째 시행: [450, 439] -> 11
50번째 시행: [457, 450] -> 7
51번째 시행: [466, 458] -> 8
52번째 시행: [473, 467] -> 6
53번째 시행: [483, 473] -> 10
54번째 시행: [493, 483] -> 10
55번째 시행: [501, 494] -> 7
56번째 시행: [511, 502] -> 9
57번째 시행: [517, 511] -> 6
58번째 시행: [527, 517] -> 10
59번째 시행: [536, 527] -> 9
60번째 시행: [547, 538] -> 9
61번째 시행: [562, 550] -> 12
62번째 시행: [593, 567] -> 26
first_max_diff=38

```

```

C:\Users#tjdgu#OneDrive#바탕 화면#buss2.exe
최적의 최대 대기시간: 5
( 3 ) 1명 / 대기시간:0
(15 ) 1명 / 대기시간:0
(24 25 29 ) 3명 / 대기시간:5
(36 40 41 ) 3명 / 대기시간:5
(43 43 45 46 47 47 ) 6명 / 대기시간:4
(52 55 56 57 ) 4명 / 대기시간:5
(58 59 ) 2명 / 대기시간:1
(64 65 68 ) 3명 / 대기시간:4
(70 71 71 73 73 73 75 ) 7명 / 대기시간:5
(76 76 76 76 77 78 79 ) 7명 / 대기시간:3
(82 84 87 87 ) 4명 / 대기시간:5
(91 93 95 96 ) 4명 / 대기시간:5
(97 97 98 101 102 ) 5명 / 대기시간:5
(103 103 103 104 105 106 106 107 ) 8명 / 대기시간:4
(108 108 109 109 110 111 111 113 ) 8명 / 대기시간:5
(114 114 114 116 116 116 116 117 ) 8명 / 대기시간:3
(117 117 118 119 ) 4명 / 대기시간:2
(124 125 125 127 127 ) 5명 / 대기시간:3
(130 131 131 131 132 133 133 133 ) 8명 / 대기시간:3

```

```

C:\Users#tjdgu#OneDrive#바탕 화면#buss2.exe
(440 ) 1명 / 대기시간:0
(446 447 449 450 450 451 ) 6명 / 대기시간:5
(452 452 453 456 457 457 ) 6명 / 대기시간:5
(458 460 463 ) 3명 / 대기시간:5
(464 465 465 465 466 467 469 469 ) 8명 / 대기시간:5
(471 471 472 473 473 473 474 476 ) 8명 / 대기시간:5
(476 479 480 480 ) 4명 / 대기시간:4
(483 483 486 487 488 488 ) 6명 / 대기시간:5
(489 490 493 494 494 494 ) 6명 / 대기시간:5
(495 495 496 500 ) 4명 / 대기시간:5
(501 502 506 ) 3명 / 대기시간:5
(507 509 510 510 511 511 511 512 ) 8명 / 대기시간:5
(513 514 515 516 517 517 517 517 ) 8명 / 대기시간:4
(518 519 519 523 ) 4명 / 대기시간:5
(524 527 527 528 528 528 529 ) 7명 / 대기시간:5
(533 535 536 538 ) 4명 / 대기시간:5
(541 542 542 543 545 ) 5명 / 대기시간:4
(547 547 550 552 ) 4명 / 대기시간:5
(554 555 558 559 ) 4명 / 대기시간:5
(561 562 ) 2명 / 대기시간:1
(567 570 ) 2명 / 대기시간:3
(591 593 ) 2명 / 대기시간:2

n:500명, 각 버스에 탄 사람들을 합친 수:500명, m:100대, 이용 버스:90대
최대 대기 시간은 5

0.109초입니다.

```

0.109초 (정렬~'최대 대기시간=5' 출력까지)

③ CMD 명령어 이용 TXT파일 출력 결과 1 (② 와 같은 입력)

```
test - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
500 100 8
41 68 36 3 73 29 84 65 70 73 15 56 93 40 75 106 111 59 45 55 111 25 24 76 116 107 47 43 46 124 77 57 103 71 103 47 103 136 73 133 43 52 64 76 117 109 87 58 101 117 97 9
89 368 327 347 364 362 349 287 276 287 281 348 341 369 283 372 339 308 288 372 370 374 378 317 309 350 326 327 365 351 319 388 376 330 370 395 329 360 340 356 349 32
```

< test.txt > 입력용 텍스트 파일 (500개 데이터, 범위:3~593)

```
out - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
(440 ) 1명 / 대기시간:0
(446 447 449 450 450 451 ) 6명 / 대기시간:5
(452 452 453 456 457 457 ) 6명 / 대기시간:5
(458 460 463 ) 3명 / 대기시간:5
(464 465 465 465 466 467 469 469 ) 8명 / 대기시간:5
(471 471 472 473 473 473 474 476 ) 8명 / 대기시간:5
(476 479 480 480 ) 4명 / 대기시간:4
(483 483 486 487 488 488 ) 6명 / 대기시간:5
(489 490 493 494 494 494 ) 6명 / 대기시간:5
(495 495 496 500 ) 4명 / 대기시간:5
(501 502 506 ) 3명 / 대기시간:5
(507 509 510 510 511 511 511 512 ) 8명 / 대기시간:5
(513 514 515 516 517 517 517 517 ) 8명 / 대기시간:4
(518 519 519 523 ) 4명 / 대기시간:5
(524 527 527 528 528 528 529 ) 7명 / 대기시간:5
(533 535 536 538 ) 4명 / 대기시간:5
(541 542 542 543 545 ) 5명 / 대기시간:4
(547 547 550 552 ) 4명 / 대기시간:5
(554 555 558 559 ) 4명 / 대기시간:5
(561 562 ) 2명 / 대기시간:1
(567 570 ) 2명 / 대기시간:3
(591 593 ) 2명 / 대기시간:2

n:500명, 각 버스에 탄 사람들을 합친 수:500명, m:100대, 아중 버스:90대
최대 대기 시간은 5

0.000초입니다.
```

< out.txt > 출력 결과 / 0.000초(입력~'최대 대기 시간=5' 출력까지)

④ CMD 명령어 이용 TXT파일 출력 결과 2 (큰 수에 적용)

```

test1 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
|00000 1000 235
751508198 187585530 4827677 540680639 481265321 656336220 283952255 33094804 977657493 190010908 555129422 351022246 842383161 271843396 116717308 12474864 2:
11 384872910 192348295 987711815 214424249 76795976 629822421 680217079 334210395 715899388 836333390 563660946 585184488 597980382 951390604 526475647 54196:
65 741448953 958988457 28813033 750930253 59538035 785510909 755946849 260951047 410214802 93556555 311512581 810212335 5799475 954532948 400409506 604156744 :
71712900 214130350 949010102 59611829 746435737 212636540 523172491 448791637 923921798 485831238 478542815 22724187 928891956 410897986 84091040 411806994 9:
62 513742045 734439811 549420071 72261001 932419377 410059542 944371200 649911168 679623392 826551946 885253043 228215886 177159562 454117236 940488306 80487:
210402817 897698952 468094499 336469232 882043816 479687194 543559307 580884731 932613531 668528640 799593681 649408002 905975265 294647671 940910351 7451002:
4050658 617710803 40629314 92781869 152511563 8658504 319504942 241558518 254918784 288730771 170599566 661168370 694170609 260086424 1726049 240149032 56797:
132935095 374576732 177182434 556821699 884553872 147861860 190318772 519137753 38998226 493957124 370086655 568155298 528991497 648396835 991482867 5542841 :
485973188 910569610 453078767 931961080 760896549 451623141 543322548 689221678 640939921 658634891 44283978 860212511 400966971 87863373 97340165 417301454 :
10510 900086732 92857126 408293494 31451924 170434167 724763049 571357113 712596040 299477471 839103420 784374579 805849807 41890783 879802136 387475297 7158:
04895 323704543 335692489 900324568 48764289 49075509 203264235 445034947 43966256 5392653 19009905 899689563 288568156 81036262 165168893 356800326 233716431
8786393 249097295 104392779 532011957 509929190 964270165 683857955 831915283 463631920 279965356 588496323 874304165 510987439 134845057 1000628 205252407 8:
433574 14793900 28882628 617925359 354303532 228650499 752305418 346502442 306423504 698680974 89033994 816937776 371004656 551597857 146573624 361271796 799:
34183 58594193 486649510 48441470 423607986 749916568 516174794 501274841 562323139 810905568 58173789 353948297 935089014 545689025 5673086 102672537 412524:
85266 502170696 299327804 35797827 775516947 223024136 9957618 947381333 641742749 95195421 43007182 56351266 815686515 186539976 963635734 435602721 1157789:
43421 551304898 646914898 334079013 974081559 741854930 678193893 111681931 627790764 595341275 338031461 635850919 321490247 288997126 859669566 96451923 71:
84921 730311449 273352 132659469 769981679 128811183 707542563 68464389 876087233 210382706 39949632 850695998 849391973 607123316 231486133 769157877 7966300:
24 641190821 749811648 611092515 645670667 646280344 660747629 359461775 930663150 274443260 930769090 215072592 367458940 100102475 53083517 960469009 40305:
58208 212867739 241493969 339246622 3363269 926439876 475941421 195053187 849081297 984715 38480007 528313726 824590596 963814105 965938899 653488309 84880511:
609273 828311232 588959255 617556913 242844265 866285099 750839314 340509835 415901055 802151116 898070940 15939416 381666119 319990301 108428037 42885604 41:
1609788 663452633 583437609 621744660 663807198 754434321 512550763 445951490 39208635 682432071 700876870 422932800 27036552 614196319 811708595 314457473 8:
39101444 565597765 780609827 977903847 713499874 884823208 341240479 391919824 30269345 517170019 564038959 241330165 151176352 799860926 371670291 606944753
53456948 256879980 354739351 779456611 352904312 972081914 420545024 824799693 146532820 181379783 41339974 227606768 871188942 6595205 148475316 344073550 1:
736 20696214 69813505 576639968 372325967 927941150 186088851 78272819 332504413 63069118 74948074 29831825 360860988 556556903 100899312 195138511 566819661
969521 513499002 863289755 901568816 558027211 755832571 456003552 933668321 49342631 158182620 308929199 140706989 661708093 127415921 644340481 230472725 8:
3201945 698436807 919874154 64461925 415136279 726778396 669187131 851300610 65660368 57161727 900726821 889640160 999920264 19615657 17257031 341520942 2343:
09 279464270 20289274 389297747 250952018 536998630 673527075 419172878 301921740 662693684 571165214 27184855 95680179 33002589 701651073 541882621 5349759:
52552 32801847 901235729 33801395 979746498 160189082 846919663 895965346 230869088 682524912 864262399 656909541 981410516 418456119 26536755 942919425 4531:
362732451 351974177 563394715 394899695 61180303 449010917 913842312 469992039 483374623 894033421 847818714 834056859 723913401 716384820 60633099 531519434 >
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```

< test1.txt > 입력용 텍스트 파일 (100,000개 데이터, 범위:1만~9억9천)

```

out - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
간:988540
(983295830 983329261 983335839 983358851 983363624 983371949 983373088 983373873 983375711 983380879 983388565 983406566 983419173 983424002 983429965 98343:
(984288963 984296299 984299844 984302530 984313552 984344695 984371988 984404125 984413573 984424670 984429408 984430274 984454530 984460439 984491574 9845008
(985291647 985292535 985298551 985322979 985345878 985357137 985358515 985379616 985406083 985408840 985409709 985413872 985420807 985423898 985427027 9854446
(986285968 986303314 986306366 986318903 986339302 986340823 986345110 986345550 986363020 986368565 986393734 986408873 986463950 986464958 986497871 9865156
(987285008 987309281 987321356 987328340 987346427 987348175 987351299 987367166 987402761 987404246 987409352 987414173 987464145 987480063 987489580 987495
(988281630 988288681 988298459 988318441 988327492 988337855 988357794 988359065 988368998 988381603 988389927 988401893 988405737 988418646 988438480 988458
(989289729 989298010 989314733 989315758 989322808 989343644 989359388 989394355 989408645 989436297 989438348 989448232 989453802 989460784 989466391 989470
(990279810 990312709 990316957 990317907 990326243 990331661 990345331 990356644 990357419 990359013 990376477 990384177 990397683 990401942 990414160 990419
1241857 991263619 ) 104명 / 대기시간:983809
(991284798 991287608 991296754 991367973 991388865 991390596 991399096 991399585 991404351 991413771 991421881 991428530 991443491 991462703 991472099 991481
(992290044 992315067 992333313 992333582 992336596 992338191 992352908 992379201 992398396 992401568 992404770 992410453 992411699 992416910 992418806 9924225
(993284331 993288633 993292845 993297835 993303973 993308705 993333537 993338196 993357634 993360709 993362707 993367697 993369029 993380152 993384772 993420
4265634 ) 103명 / 대기시간:981303
(994278657 994294386 994299842 994315249 994336657 994348795 994370170 994390782 994390973 994396723 994414211 994416197 994428872 994456364 994481273 994485
5210323 995219030 995219769 995234558 995236487 995249214 995262443 ) 109명 / 대기시간:983786
(995275731 995280085 995283229 995290166 995313934 995324950 995331435 995331467 995353789 995371336 995382828 995384799 995385004 995430274 995440513 9954449
6226700 996227611 996228339 996229414 996231797 996246716 ) 108명 / 대기시간:970985
(996269932 996295640 996305745 996324664 996332631 996345242 996348806 996357979 996365432 996371223 996383466 996383971 996391070 996391243 996394331 996428
(997267636 997306966 997325579 997329220 997346175 997357592 997398881 997407593 997408763 997417217 997427012 997469861 997482162 997504434 997524846 997526
(998272581 998273626 998287281 998328163 998331642 998340922 998345824 998361302 998372365 998376145 998377078 998388064 998396586 998399643 998403517 998403
(999274464 999275490 999288131 999307734 999314426 999315947 999337908 999340566 999345503 999390376 999406347 999410962 999411329 9994443525 999446884 999456
n:100000명, 각 버스에 탄 사람들을 합친 수:100000명, m:1000대, 이용 버스:1000대
최대 대기 시간은 989499
0.082초입니다.
Ln 5, Col 22 100% Windows (CRLF) ANSI

```

< out.txt > 출력 결과 / 0.082초(입력~'최대 대기 시간=989499' 출력까지)

전체 코드

4-1] 버스 전체 코드

① 버스 분석 코드 제외한 코드

```
#include <stdio.h>
#include <stdlib.h>

int n, m, c;

void swap_elements(int* list, int index1, int index2) {
    int temp = list[index1];
    list[index1] = list[index2];
    list[index2] = temp;
}

int partition(int* list, int start, int end) {
    int i, b;
    i = b = start;
    int p = end;

    while (i < p) {
        if (list[i] <= list[p]) {
            swap_elements(list, i, b);
            b += 1;
        }
        i += 1;
    }
    swap_elements(list, b, p);
    p = b;

    return p;
}

int quick_sort(int* list, int start, int end) {
    if (end - start < 1)
        return 0;

    int p = partition(list, start, end);
    quick_sort(list, start, p - 1);
    quick_sort(list, p + 1, end);
}

int bus_num(int* list, int max_diff) {
    int a = 0; // list[a]~list[b]는 현재 버스에 탈 수 있을지 조사할 때 필요한
    사람들(시각), list[a]~list[b-1]은 현재 버스에 탄 사람들
    int b = 1;
    int bus = 1;
```

```

while (b < n) { // list[n-1]까지 존재함
    if (list[b] - list[a] <= max_diff) { // 대기시간이 최대 대기시간
이하이다.
        if ((b - a + 1) > c) { // 사람 수가 좌석 수에 불만족
            bus++; // 버스 늘림
            a = b; // 다음 버스 처리 시작
        }
    }
    else { // 대기시간이 최대 대기시간 초과이다.
        bus++; // 버스 늘림
        if (bus > m) // 최대 버스 수 초과
            break; // 버스 조사 끝냄
        a = b; // 다음 버스 처리 시작
    }
    b++; // 사람 태움, 다음 경우 조사를 위해 b를 1증가
}
return bus;
}

int binary_search_compare(int* list, int first, int last, int fmd) {

    if ((last == 0) || (first == fmd)) // 처음 값 혹은 마지막 값 처리 (최종의 경우),
조건문에서 생략했지만 first=last임
        return first; // 최종의 경우까지 오면 나올 수밖에 없음

    int mid = (first + last) / 2;

    int bus_mid = bus_num(list, mid); // 최대 대기시간(max_diff)이 mid일 때, bus값

    if (mid == 0) { // first:0, last:1이어서 mid가 0이면 'Main Part'의 연산이 불가능
-> 따로 처리
        if (bus_mid <= m) // max_diff가 0일 때 조건 만족 -> 0이 최적의 최대
대기시간
            return 0;
        else
            return 1;
    }

    // Main Part
    if (bus_mid <= m) { // max_diff가 0이 아닌 mid일 때, bus<=m [현재 조건
만족(적거나 같음)]
        if (bus_num(list, mid - 1) > m) // max_diff가 mid일 때, bus>m [바로 전일
때 조건 불만족]
            return mid; // best_time은 mid

        else // [바로 전일 때 조건 만족]
            binary_search_compare(list, first, mid - 1, fmd); // '이전 값들
중 하나'에서 재조사
    }
    else // max_diff가 0이 아닌 mid일 때, bus>m [현재 조건 불만족]
        binary_search_compare(list, mid + 1, last, fmd); // '다음 값들 중
하나'에서 재조사
}

```

```

}

int main() {
    int i; // for iteration

    // input processing
    scanf("%d %d %d", &n, &m, &c);

    // find error
    if (n < c) {
        printf("[n<c] error");
        return 0;
    }
    if (m * c < n) {
        printf("[mc<n] error");
        return 0;
    }
    // end
    if (n <= 1 || c == 1) {
        printf("0");
        return 0;
    }

    int* time_list = (int*)malloc(sizeof(int) * n);

    for (i = 0; i < n; i++)
        scanf("%d", &time_list[i]);

    // sorting
    quick_sort(time_list, 0, n - 1);

    // make first max_diff (첫 번째 최대 대기시간(최대 대기차))
    int fmd = 0;
    int tmp = 0;
    int idx = 0; // index of time_list
    int M = m; // temp_m

    if ((n % c) == 0) { // n=m*c, n<m*c (n = k*c)
        if ((m * c - n) >= c) // ----- (m-α) buses can go
            M = n / c;
        for (i = 0; i < M; i++) {
            tmp = time_list[idx + (c - 1)] - time_list[idx];
            idx += c;
            //find max
            if (tmp >= fmd)
                fmd = tmp;
        }
    }
    else { // n<m*c (n != k*c)
        if ((m * c - n) >= c) // ----- (m-α) buses can go
            M = (n / c) + 1;
    }
}

```

```

        for (i = 0; i < M; i++) {
            if (i == M - 1) {
                tmp = time_list[idx + ((n % c) - 1)] - time_list[idx];
            }
            else {
                tmp = time_list[idx + (c - 1)] - time_list[idx];
                idx += c;
            }
            //find max
            if (tmp >= fmd)
                fmd = tmp;
        }
    }

    // find best max_diff
    int best_wait = binary_search_compare(time_list, 0, fmd, fmd); //
(리스트, first, last, fmd)
    printf("%d", best_wait);

    return 0;
}

```