|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』과제 보고서 | | | |
| 제목 | 1장 실습( ) 과제( O ) | 제출일자 | 2022.   09.  18 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 연습문제 1장  02)  INPUT = A, B  OUTPUT MAX(A,B)  MAX(A, B)  if A > B then  return A  return B |
| 04)  #include <stdio.h>  #include <time.h>  #include <stdlib.h>  #include <stdbool.h>  typedef struct Node {  int v;  struct Node\* next;  } Node;  typedef struct Set {  Node\* head;  bool (\*Is\_In) (struct Set\* list, int n);  void (\*Create) (struct Set\* list, int v);  void (\*Insert) (struct Set\* list, int v);  bool (\*Remove) (struct Set\* list, int v);  } Set;  void printSet(Set\* list);  void init(Set\* list);  Set Union(Set\* A, Set\* B);  Set Intersection(Set\* A, Set\* B);  Set Difference(Set\* A, Set\* B);  int main() {  srand(time(NULL));  int A[10], B[10];  int i;  Set set\_A, set\_B;  init(&set\_A), init(&set\_B);  printf("======= Random Array A =======\n");  for (i = 0;i < 10;i++) {  A[i] = rand() % 10;  B[i] = rand() % 10;  i < 9 ? printf("%d, ", A[i]) : printf("%d\n", A[i]);  }  printf("======= Random Array B =======\n");  for (i = 0;i < 10;i++) {  i < 9 ? printf("%d, ", B[i]) : printf("%d\n\n", B[i]);  set\_A.Create(&set\_A, A[i]);  set\_B.Create(&set\_B, B[i]);  }  Set set\_Union = Union(&set\_A, &set\_B);  Set set\_Intersection = Intersection(&set\_A, &set\_B);  Set set\_Diff\_AB = Difference(&set\_A, &set\_B);  Set set\_Diff\_BA = Difference(&set\_B, &set\_A);  printf("============ SET A ===========\n");  printSet(&set\_A);  printf("============ SET B ===========\n");  printSet(&set\_B);  printf("\n");  printf("== INSERT 10 to setA 2 loop ==\n");  for (i = 0; i != 2; i++) {  if (set\_A.Is\_In(&set\_A, 10)) printf("이미 존재하는 값\n");  else {  set\_A.Insert(&set\_A, 10);  printSet(&set\_A);  }  }  printf("\n");  printf("== REMOVE 10 to setA 2 loop ==\n");  for (i = 0; i != 2; i++) {  if (set\_A.Is\_In(&set\_A, 10)) {  set\_A.Remove(&set\_A, 10);  printSet(&set\_A);  }  else printf("존재하지 않는 값\n");  }  printf("\n");  printf("============ UNION ===========\n");  printSet(&set\_Union);  printf("\n");  printf("======== Intersection ========\n");  printSet(&set\_Intersection);  printf("\n");  printf("======= Difference A-B =======\n");  printSet(&set\_Diff\_AB);  printf("======= Difference B-A =======\n");  printSet(&set\_Diff\_BA);  printf("\n");  return 0;  }  void printSet(Set\* list) {  if (list -> head == NULL) printf("empty\n");  else {  Node\* ptr = list->head;  if (ptr != NULL) printf("%d", ptr->v);  ptr = ptr->next;  while (ptr != NULL) {  printf(", %d", ptr->v);  ptr = ptr->next;  }  printf("\n");  }  }  bool Set\_Is\_In(Set\* list, int n) {  Node\* ptr = list->head;  while (ptr != NULL) {  if (ptr->v == n) {  return true;  }  ptr = ptr->next;  }  return false;  }  void push(Set\* list, int v) {  if (Set\_Is\_In(list, v)) return;  Node\* ptr = list->head;  Node\* node = (Node\*)malloc(sizeof(Node));  node->v = v;  node->next = NULL;  if (ptr == NULL) list->head = node;  else {  while (ptr->next != NULL) ptr = ptr->next;  ptr->next = node;  }  }  Set Intersection(Set\* A, Set\* B) {  Set set;  init(&set);  Node\* ptr = B->head;  while (ptr != NULL) {  if (A->Is\_In(A, ptr->v)) set.Create(&set, ptr->v);  ptr = ptr->next;  }  return set;  }  Set Difference(Set\* A, Set\* B) {  Set set;  init(&set);  Node\* ptr = A->head;  while (ptr != NULL) {  if (!(B->Is\_In(B,ptr->v))) set.Create(&set, ptr->v);  ptr = ptr->next;  }  return set;  }  Set Union(Set\* A, Set\* B) {  Set set;  init(&set);  Node\* ptr = A->head;  while (ptr != NULL) {  set.Create(&set, ptr->v);  ptr = ptr->next;  }  ptr = B->head;  while (ptr != NULL) {  set.Create(&set, ptr->v);  ptr = ptr->next;  }  return set;  }  bool pop(Set\* list, int v) {  Node\* ptr = list->head;  Node\* next = ptr->next;  if (ptr == NULL) return false;  if (ptr->v == v) {  list->head = ptr->next;  return true;  }  while (next != NULL) {  if (next->v == v) {  ptr->next = next->next;  return true;  }  ptr = ptr->next;  next = next->next;  }  return false;  }  void init(Set\* list) {  list->head = NULL;  list->Is\_In = Set\_Is\_In;  list->Create = push;  list->Insert = push;  list->Remove = pop;  }  실행결과 |
| 06)  for(i=1; i<n; i\*=2) printf(”Hello”);  출력을 n까지 만큼 증가 하므로 n이 8이면 3번 실행된다.  즉 O(logN)이다. |
| 08)  n = 1024 라고 가정,  n^2+10n+8 = 1048576+10240+8 = 1058824  O(n) = O(1024), O(nlogn) = O(10240),  O(n^2) = O(1048576), O(n^2logn) = O(10485760)  ∴ n의 값이 증가할수록 n^2의 근사 값이란 것을 알 수 있다. O(n^2) |
| 10)  n 을 100이라고 가정  O(n^2) = O(10000), n값을 2배 = 200  O(200^2) = O(40000), 따라서 입력의 값이 2배 증가하면 실행시간은 4배 증가함을 알 수 있다. |
| 12)  n을 1024라고 가정  1. O(1) = 1번, 2. O(N) = 1024번, 3. O(logN) = 10번, 4. O(N^2) = 1048576번  5. O(NlogN) = 10240번, 6. O(N!) = overflow, 7. O(2^N) = overflow  6과 7을 비교하기 위해 N을 10으로 수정  O(10!) = 3628800, O(2^10) = 1024  N의 값이 증가할수록  O(1) -> O(logN) -> O(N) -> O(NlogN) -> O(N^2) -> O(2^N) -> O(N!)  순서로 수행시간이 적게걸린다. |
| 14)   |  |  | | --- | --- | | 입력의 개수 N | 수행시간(초) | | 2 | 2 | | 4 | 8 | | 8 | 25 | | 16 | 63 | | 32 | 162 |   N = 2 일 때, 수행시간이 2초일 경우 O(N), O(NlogN), O(N!)  N = 4 일 때, 수행시간이 8초일 경우 O(NlogN)  2번과 4번으로 빅오 표기법이 O(NlogN)임을 알 수 있다.  검산  N = 8, 8\*3 = 24  N = 16, 16\*4 = 64  N = 32, 32\*5 = 160  수행시간들이 O(NlogN)에 상한함이 보인다.  따라서 빅오표기법으로 나타낼 경우 O(NlogN)이다. |
| 16)  빅오 표기법의 정의에 따라서,  f(n)=6n^2+3n, g(n)=n 이라고 하면  n=2 일 때 |f(n)| ≤ c|g(n)| 를 만족하는 상수 c는 31이므로  f(n)=6n^2+3n <31∙g(n)=31n 라는 식이 나오지만 n이 2를 초과할 경우 성립하지 않는다.  따라서 6n^2 + 3n은 O(n)이 될 수 없다. |