**SẮP XẾP VÀ TÌM KIẾM**

|  |  |
| --- | --- |
| Sắp xếp nổi bọt (Bubble Sort)  Ý tưởng:  So sánh và hoán đổi: Duyệt qua mảng từ đầu đến cuối, so sánh từng cặp phần tử liền kề và hoán đổi chúng nếu chúng không theo thứ tự.  Lặp lại: Lặp lại quá trình cho đến khi mảng được sắp xếp hoàn toàn. Sau mỗi lần lặp, phần tử lớn nhất trong đoạn chưa sắp xếp sẽ "nổi" lên cuối mảng. | void bubbleSort(int arr[], int n) {    for (int i = 0; i < n - 1; i++) {      for (int j = 0; j < n - i - 1; j++) {        if (arr[j] > arr[j + 1]) {          int temp = arr[j];          arr[j] = arr[j + 1];          arr[j + 1] = temp;        }      }    }  } |
| Sắp xếp chèn (Insertion Sort)  Ý tưởng:  Chèn phần tử vào đúng vị trí: Duyệt qua mảng từ trái sang phải, tại mỗi phần tử, chèn nó vào đúng vị trí trong đoạn đã được sắp xếp bên trái.  Lặp lại: Lặp lại quá trình cho đến khi tất cả các phần tử được chèn vào đúng vị trí. | void insertSort(int arr[], int n) {    for (int i = 1; i < n; i++) {      int key = arr[i];      int j = i - 1;      while (j >= 0 && arr[j] > key) {        arr[j + 1] = arr[j];        j--;      }      arr[j + 1] = key;    }  } |
| Sắp xếp chọn (Selection Sort)  Ý tưởng:  Tìm phần tử nhỏ nhất: Trong mỗi lần duyệt qua mảng, tìm phần tử nhỏ nhất và hoán đổi nó với phần tử đầu tiên của mảng chưa sắp xếp.  Lặp lại: Tiếp tục quá trình này với phần tử thứ hai, thứ ba,... cho đến khi mảng được sắp xếp hoàn toàn. | void selectionSort(int arr[], int n) {    for (int i = 0; i < n - 1; i++) {      int minIndex = i;      for (int j = i + 1; j < n; j++) {        if (arr[j] < arr[minIndex]) {          minIndex = j;        }      }      int temp = arr[i];      arr[i] = arr[minIndex];      arr[minIndex] = temp;    }  } |
| Sắp xếp vun đống (Heap Sort)  Ý tưởng:  Xây dựng Heap: Chuyển mảng thành một cây nhị phân dạng heap (thường là max-heap) sao cho phần tử lớn nhất là gốc của cây.  Sắp xếp: Hoán đổi phần tử gốc với phần tử cuối cùng của mảng chưa sắp xếp, sau đó giảm kích thước của heap đi 1 và điều chỉnh lại heap để phần tử lớn nhất lại trở thành gốc. Lặp lại quá trình cho đến khi mảng được sắp xếp. | void heapify(int arr[], int n, int i) {    int largest = i;    int left = 2 \* i + 1;    int right = 2 \* i + 2;    if (left < n && arr[left] > arr[largest]) {      largest = left;    }    if (right < n && arr[right] > arr[largest]) {      largest = right;    }    if (largest != i) {      int temp = arr[i];      arr[i] = arr[largest];      arr[largest] = temp;      heapify(arr, n, largest);    }  }  void heapSort(int arr[], int n) {    for (int i = n / 2 - 1; i >= 0; i--) {      heapify(arr, n, i);    }    for (int i = n - 1; i >= 0; i--) {      int temp = arr[0];      arr[0] = arr[i];      arr[i] = temp;      heapify(arr, i, 0);    }  } |
| Sắp xếp trộn (Merge Sort)  Ý tưởng:  Chia nhỏ mảng: Chia mảng thành hai mảng con bằng cách chia đôi mảng ban đầu.  Đệ quy: Áp dụng đệ quy để tiếp tục chia nhỏ mảng con cho đến khi mỗi mảng con chỉ còn một phần tử.  Trộn: Trộn hai mảng con đã được sắp xếp thành một mảng lớn hơn đã được sắp xếp. | void merge(int arr[], int l, int m, int r) {    int n1 = m - l + 1;    int n2 = r - m;    int L[n1], R[n2];    for (int i = 0; i < n1; i++) {      L[i] = arr[l + i];    }    for (int j = 0; j < n2; j++) {      R[j] = arr[m + 1 + j];    }    int i = 0;    int j = 0;    int k = l;    while (i < n1 && j < n2) {      if (L[i] <= R[j]) {        arr[k] = L[i];        i++;      } else {        arr[k] = R[j];        j++;      }      k++;    }    while (i < n1) {      arr[k] = L[i];      i++;      k++;    }    while (j < n2) {      arr[k] = R[j];      j++;      k++;    }  }  void mergeSort(int arr[], int l, int r) {    if (l >= r) {      return;    }    int m = l + (r - l) / 2;    mergeSort(arr, l, m);    mergeSort(arr, m + 1, r);    merge(arr, l, m, r);  } |
| Sắp xếp nhanh (Quick Sort)  Ý tưởng:  Chọn điểm chốt: Chọn một phần tử làm điểm chốt.  Phân chia: Sắp xếp lại mảng sao cho tất cả các phần tử nhỏ hơn điểm chốt đứng bên trái, và tất cả các phần tử lớn hơn hoặc bằng điểm chốt đứng bên phải.  Đệ quy: Đệ quy áp dụng quá trình trên cho mảng con bên trái và bên phải của điểm chốt cho đến khi mảng con có kích thước 0 hoặc 1. | void quickSort(int arr[], int low, int high) {    if (low < high) {      int pivot = arr[high];      int i = low - 1;      for (int j = low; j < high; j++) {        if (arr[j] < pivot) {          i++;          int temp = arr[i];          arr[i] = arr[j];          arr[j] = temp;        }      }      int temp = arr[i + 1];      arr[i + 1] = arr[high];      arr[high] = temp;      int pi = i + 1;      quickSort(arr, low, pi - 1);      quickSort(arr, pi + 1, high);    }  } |

|  |  |
| --- | --- |
| int binarySearch(int arr[], int n, int x) {    int left = 0, right = n - 1;    while (left <= right) {      int mid = left + (right - left) / 2;      if (arr[mid] == x) {        return mid;      }      if (arr[mid] < x) {        left = mid + 1;      } else {        right = mid - 1;      }    }    return -1;  } | int linearSearch(int arr[], int n, int x) {    for (int i = 0; i < n; i++) {      if (arr[i] == x) {        return i;      }    }    return -1;  } |