Домашнє завдання №10

Для алгоритму, що заданий за допомогою функції мовою С, розрахувати обчислювальну складність для «найгіршого випадку» та сформувати блок-схему алгоритму.

** писати альтернативний коду не є обов'язково*

*** компілювати і запускати код на виконання не є обов'язково*

Вибір варіанту

(Nж + Nг + 1) % 4+ 1
де: Nж – порядковий номер студента в групі, а Nг – номер групи(1,2,3,4,5,6,7 або 8)

Варіанти завдання

Варіант	Код
1	**Сортування вибором**

	void choiceSort(int * array, int leftIndex, int rightIndex){
	for(int iIndex = leftIndex; iIndex <= rightIndex; ++iIndex){
	int kIndex = iIndex;
	int temp = array[iIndex];
	int exch = 0;
	
	for(int jIndex = iIndex + 1; jIndex <= rightIndex; ++jIndex){
	if(array[jIndex] < temp){
	kIndex = jIndex;
	temp = array[jIndex];
	exch = 1;
	}
	}
	
	if(exch){
	array[kIndex] = array[iIndex];
	array[iIndex] = temp;
	}
	}
	}

2	**Сортування включенням**

	void insertSort(int * array, int leftIndex, int rightIndex){
	for(int iIndex = leftIndex + 1; iIndex <= rightIndex; ++iIndex){
	int temp = array[iIndex];
	int jIndex = iIndex - 1;
	for(; jIndex >= leftIndex && array[jIndex] > temp; --jIndex){
	array[jIndex + 1] = array[jIndex];
	}
	array[jIndex + 1] = temp;
	}
	}

3	**Сортування Шелла**

	`void shellSort(int * array, int leftIndex, int rightIndex){`
	`  int sortingSize =  rightIndex - leftIndex + 1;`
	`  for (int dIndex = leftIndex + sortingSize/2; dIndex >= leftIndex + 1; dIndex /= 2){`
	`  for (int iIndex = dIndex; iIndex < sortingSize; iIndex++){`
	`  for (int jIndex = iIndex; jIndex >= dIndex && array[jIndex - dIndex] > array[jIndex]; jIndex -= dIndex){`
	`  int temp = array[jIndex];`
	`  array[jIndex] = array[jIndex - dIndex];`
	`  array[jIndex - dIndex] = temp;`
	`  }`
	`  }`
	`  }`
	`}`

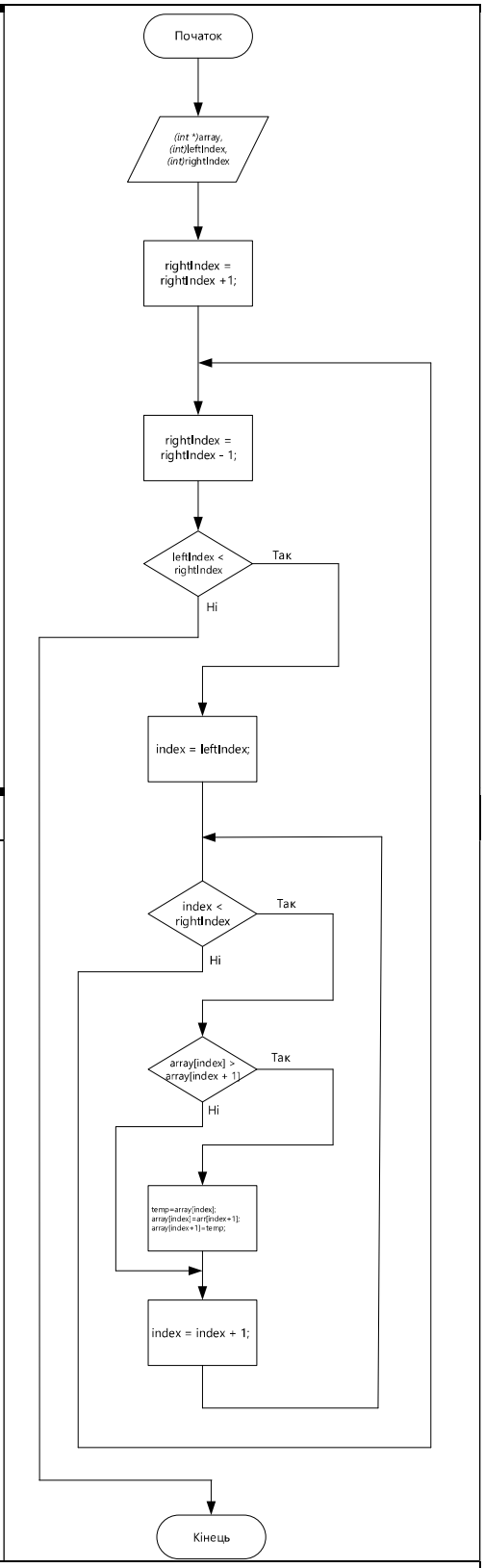
4	**Швидке сортування**

	`void quickSort(int * array, int leftIndex, int rightIndex){`
	`  int iIndex = leftIndex;`
	`  int jIndex = rightIndex;`
	`  int xElemntValue = array[(leftIndex + rightIndex) / 2];`
	
	`  do{`
	`  while((array[iIndex] < xElemntValue) && (iIndex < rightIndex)) {`
	`  ++iIndex;`
	`  }`
	`  while((xElemntValue < array[jIndex]) && (jIndex > leftIndex)) {`
	`  --jIndex;`
	`  }`
	`  if(iIndex <= jIndex){`
	`  int temp = array[iIndex];`
	`  array[iIndex] = array[jIndex];`
	`  array[jIndex] = temp;`
	`  ++iIndex;`
	`  --jIndex;`
	`  }`
	`  }while(iIndex <= jIndex);`
	
	`  if(leftIndex < jIndex) {`
	`  quickSort(array, leftIndex, jIndex);`
	`  }`
	
	`  if(iIndex < rightIndex) {`
	`  quickSort(array, iIndex, rightIndex);`
	`  }`
	`}`

Приклад виконання завдання

Показано виконання завдання для бульбашкового сортування.

```
void bulbSort(int * array, int leftIndex, int rightIndex){
    if(!array){
        return;
    }
    ++rightIndex;
    while(leftIndex < --rightIndex){
        for(unsigned int index = leftIndex; index <
rightIndex; ++index){
            if(array[index] > array[index + 1]){
                int temp = array[index];
                array[index] = array[index + 1];
                array[index + 1] = temp;
            }
        }
    }
}
```



Альтернативний код

```
void bulbSort1(int * array, int leftIndex, int rightIndex){
    if(!array){
        return;
    }
    for(int index = leftIndex; index < rightIndex ?
rightIndex : (index = leftIndex, --rightIndex); ++index){
        if(array[index] > array[index + 1]){
            array[index] ^= array[index + 1];
            array[index + 1] ^= array[index];
            array[index] ^= array[index + 1];
        }
    }
}
```

Аналіз «найгіршого випадку»

$$A(N) = \sum_{i=1}^{N-1}(N-i) = N(N-1) - \sum_{i=1}^{N-1} i = N(N-1) - \frac{N(N-1)}{2} = \frac{N(N-1)}{2}$$

$O(n^2)$

Приклад коду

Лістинг

```
#include <stdio.h>
#include <stdlib.h>
#define DATA_SIZE 32

#define sort bulbSort
//#define sort choiceSort
//#define sort insertSort
//#define sort shellSort
//#define sort quickSort

void bulbSort(int * array, int leftIndex, int rightIndex){
	int index, temp;

	if (!array){
		return;
	}
	++rightIndex;
	while (leftIndex < --rightIndex){
		for (index = leftIndex; index < rightIndex; ++index){
			if (array[index] > array[index + 1]){
				temp = array[index];
				array[index] = array[index + 1];
				array[index + 1] = temp;
			}
		}
	}
}

void bulbSort1(int * array, int leftIndex, int rightIndex){
	int index;

	if (!array){
		return;
	}
	for (index = leftIndex; index < rightIndex ? rightIndex : (index = leftIndex, --
rightIndex); ++index){
		if (array[index] > array[index + 1]){
			array[index] ^= array[index + 1];
			array[index + 1] ^= array[index];
			array[index] ^= array[index + 1];
		}
	}
}

void choiceSort(int * array, int leftIndex, int rightIndex){
	int iIndex, jIndex;

	for (iIndex = leftIndex; iIndex <= rightIndex; ++iIndex){
		int kIndex = iIndex;
		int temp = array[iIndex];
		int exch = 0;
		for (jIndex = iIndex + 1; jIndex <= rightIndex; ++jIndex){
			if (array[jIndex] < temp){
				kIndex = jIndex;
				temp = array[jIndex];
				exch = 1;
			}
		}
		if (exch){
			array[kIndex] = array[iIndex];
```

```
                        array[iIndex] = temp;
                }
        }
}

void insertSort(int * array, int leftIndex, int rightIndex){
        int iIndex, jIndex;

        for (iIndex = leftIndex + 1; iIndex <= rightIndex; ++iIndex){
                int temp = array[iIndex];
                jIndex = iIndex - 1;
                for (; jIndex >= leftIndex && array[jIndex] > temp; --jIndex){
                        array[jIndex + 1] = array[jIndex];
                }
                array[jIndex + 1] = temp;
        }
}

void shellSort(int * array, int leftIndex, int rightIndex){
        int dIndex, iIndex, jIndex;

        int sortingSize = rightIndex - leftIndex + 1;
        for (dIndex = leftIndex + sortingSize / 2; dIndex >= leftIndex + 1; dIndex /= 2){
                for (iIndex = dIndex; iIndex < sortingSize; iIndex++){
                        for (jIndex = iIndex; jIndex >= dIndex && array[jIndex - dIndex] >
                                array[jIndex]; jIndex -= dIndex){
                                int temp = array[jIndex];
                                array[jIndex] = array[jIndex - dIndex];
                                array[jIndex - dIndex] = temp;
                        }
                }
        }
}

void quickSort(int * array, int leftIndex, int rightIndex){
        int iIndex = leftIndex;
        int jIndex = rightIndex;
        int xElemntValue = array[(leftIndex + rightIndex) / 2];
        do{
                while ((array[iIndex] < xElemntValue) && (iIndex < rightIndex)) {
                        ++iIndex;
                }
                while ((xElemntValue < array[jIndex]) && (jIndex > leftIndex)) {
                        --jIndex;
                }
                if (iIndex <= jIndex){
                        int temp = array[iIndex];
                        array[iIndex] = array[jIndex];
                        array[jIndex] = temp;
                        ++iIndex;
                        --jIndex;
                }
        } while (iIndex <= jIndex);
        if (leftIndex < jIndex) {
                quickSort(array, leftIndex, jIndex);
        }
        if (iIndex < rightIndex) {
                quickSort(array, iIndex, rightIndex);
        }
}

void printVector(void * data, int count/* 0 - full DATA_SIZE*/){
        int index = 0;
        for (index = 0; (!count || index < count) && index < DATA_SIZE; index++){
                printf("%d ", ((int *)data)[index]);
```

```
        }
        printf("\n");
}
int main(void){
        int data[DATA_SIZE] = { 1, 20, 29, 28, 10, 26, 25, 24,
                23, 22, 21, 30, 19, 18, 17, 16,
                15, 14, 13, 12, 11, 27, 9, 8,
                7, 6, 5, 4, 3, 2, 0, 31 };
        sort(data, 0, DATA_SIZE - 1);
        printVector(data, 0);
        (void)getchar();
        return EXIT_SUCCESS;
}
```