|  |
| --- |
| Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №5  по курсу «Логика и основы алгоритмизации в инженерных задачах» на тему «Обход графа в ширину» |
|  |
|  |
| Выполнила студентка группы 19ВВ3:  Ланцов А.С.  Принял:  Митрохин М. А. |
| Пенза  2020 |

### **Для матричной формы**

Листинг:

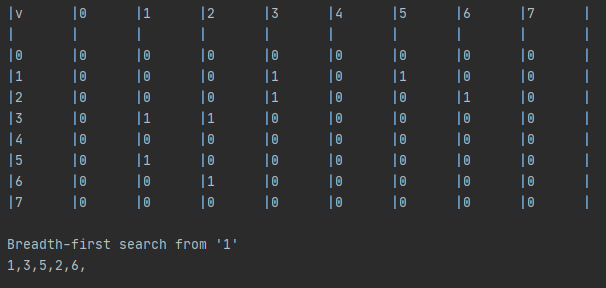
#define \_VALUE\_OF(type,pointer) \*((type\*)pointer)  
#define new(type) malloc(sizeof(type))  
#define new\_const(type,value) &((type){value})  
#define new\_block(type,how\_much) malloc(sizeof(type)\*how\_much)  
#define del(ptr) free(ptr)  
  
#define \_FOR\_(index,range) for(int index=0;index<range;index++)

typedef struct numbers\_s {  
 int\* data;  
 size\_t size;  
}numbers\_t;

numbers\_t\* numbers(int size){  
 numbers\_t\* arr = new(numbers\_t);  
 arr->data = new\_block(int,size);  
 arr->size = size;  
  
 return arr;  
}

void fill\_num(numbers\_t\* arr,int value){  
 \_FOR\_(i,arr->size){  
 arr->data[i]=value;  
 }  
}

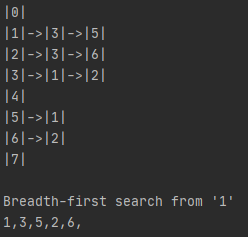
void bfs\_matrix(mtrx\_grph\_t\* graph,int vertex\_index,numbers\_t\* dist){  
 //every i in dist = -1  
 fill\_num(dist,-1);  
  
 list\_t\* vertexes\_queue = list\_create();  
  
 que\_insert(vertexes\_queue,new(int));  
 \_VALUE\_OF(int,vertexes\_queue->head->value) = vertex\_index;  
  
 int v;//next vertex  
  
 dist->data[vertex\_index] = 0;  
  
 int\* front;  
  
 printf("%d,",vertex\_index);  
 while (!que\_isempty(vertexes\_queue)){  
 front = que\_remove(vertexes\_queue);  
 v = \_VALUE\_OF(int,front);  
 del(front);  
  
 \_FOR\_(i,graph->size){  
 if (graph->matrix[v][i] && dist->data[i]==-1) {  
 que\_insert(vertexes\_queue, new(int));  
 \_VALUE\_OF(int,vertexes\_queue->tail->value) = i;  
  
 dist->data[i]=dist->data[v]+1;  
 printf("%d,",i);  
 }  
 }  
 }  
 printf("\n\n");  
 free(vertexes\_queue);  
}

Результат:  


Листовая форма:

Листинг:

void bfs\_lists(list\_graph\_t\* graph,int vertex\_index,numbers\_t\* dist){  
 fill\_num(dist,-1);  
 list\_t\* vertexes\_queue = list\_create();  
 que\_insert(vertexes\_queue,&vertex\_index);  
  
 int v,u;//next vertex  
 node\_t\* next\_vrtx;  
 dist->data[vertex\_index] = 0;  
 int\* front;  
  
 printf("%d,",vertex\_index);  
 while (!que\_isempty(vertexes\_queue)){  
 front = que\_remove(vertexes\_queue);  
 v = \_VALUE\_OF(int,front);  
  
 next\_vrtx = graph->lists[v]->head;  
  
 while(next\_vrtx->next){  
 next\_vrtx = next\_vrtx->next;  
 u = \_VALUE\_OF(int,next\_vrtx->value);  
 if(dist->data[u]==-1){  
 que\_insert(vertexes\_queue, next\_vrtx->value);  
 dist->data[u]=dist->data[v]+1;  
 printf("%d,",u);  
 }  
  
 }  
  
 }  
 printf("\n\n");  
 free(vertexes\_queue);  
}

Результат:  


Структуры:  
typedef struct node{  
 uint8\_t\* value;  
  
 struct node\* last;  
 struct node\* next;  
}node\_t;  
  
typedef struct list{  
 node\_t\* head;  
 node\_t\* tail;  
  
 int depth;  
  
}list\_t;

typedef struct matrix\_graph{  
 int\*\* src;  
  
 int\*\* matrix;  
 char\*\* vertexes;  
 int size;  
  
}mtrx\_grph\_t;

typedef struct list\_graph{  
 list\_t\*\* lists;  
 uint8\_t\*\* vertexes;  
  
 int size;  
}list\_graph\_t;

Очередь:

int que\_isempty(list\_t\* queue){  
 if(queue->head==0)  
 return 1;  
 else  
 return 0;  
}

void que\_insert(list\_t\* queue, void\* value) {  
 queue->tail = push(queue->tail,value);  
 if(queue->head==0) {  
 queue->head = queue->tail;  
 }  
  
 queue->depth ++;  
}  
  
void\* que\_remove(list\_t\* queue) {  
 if(que\_isempty(queue)) {  
 return 0;  
 }  
 else{  
 node\_t\* temp;  
 void\* head\_value;  
  
 head\_value = queue->head->value;  
 temp = queue->head;  
 queue->head = queue->head->next;  
  
  
 if(!queue->head){//if queue is empty now  
 queue->tail=NULL;  
 }  
 else{  
 queue->head->last = NULL;  
 }  
  
 free(temp);  
  
 queue->depth --;  
 return head\_value;  
 }  
  
}

**Сравнение библиотечного класса Queue и собственного:**

Листинг алгоритма на библиотечной очереди:

void std\_queue\_bfs(matrix\_graph\_s\* graph, int vertex\_index){  
 //every i in dist = -1  
 int\* dist = new int[graph->order];  
 \_FOR\_(i,graph->order){dist[i]=-1;}  
  
 queue<int> vertexes\_queue;  
 vertexes\_queue.push(vertex\_index);  
 dist[vertex\_index] = 0;  
  
 int v;//next vertex  
  
 printf("%d,",vertex\_index);  
 while (!vertexes\_queue.empty()){  
 v = vertexes\_queue.front();  
 vertexes\_queue.pop();  
  
 \_FOR\_(i,graph->order){  
 if (graph->matrix[v][i] && dist[i]==-1) {  
 vertexes\_queue.push(i);  
  
 dist[i]=dist[v]+1;  
 //printf("%d,",i);  
 }  
 }  
 }  
  
 delete[] dist;  
 printf("\n\n");  
}

|  |  |  |
| --- | --- | --- |
| Размер графа | Custom, s | Std, s |
| 1000 | 0.009 | 0.007 |
| 2000 | 0,029 | 0.021 |
| 4000 | 0,087 | 0.081 |
| 6000 | 0.173 | 0.184 |
| 8000 | 0.319 | 0.327 |

Вывод: Время работы почти не зависит от реализации структуры данных Queue.