Introduction_to_Algorithms

Generated by Doxygen 1.8.10

Sun Apr 17 2016 13:34:34

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 anonymous_namespace{adjlistgraph_test.h} Namespace Reference

Variables

- const int ADJ_NUM =10

5.1.1 Variable Documentation

5.1.1.1 const int anonymous_namespace{adjlistgraph_test.h}::ADJ_NUM =10

Definition at line 25 of file adjlistgraph_test.h.

5.2 anonymous_namespace{bellmanford_test.h} Namespace Reference

Variables

- const int B_NUM =10

5.2.1 Variable Documentation

5.2.1.1 const int anonymous_namespace{bellmanford_test.h}::B_NUM =10

Definition at line 32 of file bellmanford_test.h.

5.3 anonymous_namespace{bfs_test.h} Namespace Reference

Variables

- const int BFS_N = 10

5.3.1 Variable Documentation

5.3.1.1 const int anonymous_namespace{bfs_test.h}::BFS_N = 10

Definition at line 26 of file bfs_test.h.

5.4 anonymous_namespace{connectedcomponent_test.h} Namespace Reference

Variables

- const int C_NUM =10

5.4.1 Variable Documentation

5.4.1.1 const int anonymous_namespace{connectedcomponent_test.h}::C_NUM =10

Definition at line 32 of file connectedcomponent_test.h.

5.5 anonymous_namespace{dagshortpath_test.h} Namespace Reference

Variables

- const int DSP_NUM =10

5.5.1 Variable Documentation

5.5.1.1 const int anonymous_namespace{dagshortpath_test.h}::DSP_NUM =10

Definition at line 30 of file dagshortpath_test.h.

5.6 anonymous_namespace{dfs_test.h} Namespace Reference

Variables

- const int DFS_N = 10

5.6.1 Variable Documentation

5.6.1.1 const int anonymous_namespace{dfs_test.h}::DFS_N = 10

Definition at line 27 of file dfs_test.h.

5.7 anonymous_namespace{dijkstra_test.h} Namespace Reference

Variables

- const int DIJK_NUM =10

5.7.1 Variable Documentation

5.7.1.1 const int anonymous_namespace{dijkstra_test.h}::DIJK_NUM =10

Definition at line 31 of file dijkstra_test.h.

5.8 anonymous_namespace{disjointset_test.h} Namespace Reference

Variables

- const int S_NUM =20

5.8.1 Variable Documentation

5.8.1.1 const int anonymous_namespace{disjointset_test.h}::S_NUM =20

Definition at line 25 of file disjointset_test.h.

5.9 anonymous_namespace{floyd_warshall_test.h} Namespace Reference

Variables

- const int FW_N = 5

5.9.1 Variable Documentation

5.9.1.1 const int anonymous_namespace{floyd_warshall_test.h}::FW_N = 5

Definition at line 32 of file floyd_warshall_test.h.

5.10 anonymous_namespace{fordfulkerson_test.h} Namespace Reference

Variables

- const int FF_N = 6

5.10.1 Variable Documentation

5.10.1.1 const int anonymous_namespace{fordfulkerson_test.h}::FF_N = 6

Definition at line 31 of file fordfulkerson_test.h.

5.11 anonymous_namespace{front_flow_vertex_test.h} Namespace Reference

Variables

- const int FFV_NUM =5

5.11.1 Variable Documentation

5.11.1.1 const int anonymous_namespace{front_flow_vertex_test.h}::FFV_NUM =5

Definition at line 28 of file front_flow_vertex_test.h.

5.12 anonymous_namespace{genericpushrelabel_test.h} Namespace Reference

Variables

- const int PR_N = 6

5.12.1 Variable Documentation

5.12.1.1 const int anonymous_namespace{genericpushrelabel_test.h}::PR_N = 6

Definition at line 35 of file genericpushrelabel_test.h.

5.13 anonymous_namespace{graph_test.h} Namespace Reference

Variables

- const int G_N = 10

5.13.1 Variable Documentation

5.13.1.1 const int anonymous_namespace{graph_test.h}::G_N = 10

Definition at line 28 of file graph_test.h.

5.14 anonymous_namespace{johnson_test.h} Namespace Reference

Variables

- const int JS_N = 5

5.14.1 Variable Documentation

5.14.1.1 const int anonymous_namespace{johnson_test.h}::JS_N = 5

Definition at line 32 of file johnson_test.h.

5.15 anonymous_namespace{kruskal_test.h} Namespace Reference

Variables

- const int K_NUM =10

5.15.1 Variable Documentation

5.15.1.1 const int anonymous_namespace{kruskal_test.h}::K_NUM =10

Definition at line 30 of file kruskal_test.h.

5.16 anonymous_namespace{matrix_shortest_path_test.h} Namespace Reference

Variables

- const int MT_N = 5

5.16.1 Variable Documentation

5.16.1.1 const int anonymous_namespace{matrix_shortest_path_test.h}::MT_N = 5

Definition at line 34 of file matrix_shortest_path_test.h.

5.17 anonymous_namespace{matrixgraph_test.h} Namespace Reference

Variables

- const int MTXNUM =10

5.17.1 Variable Documentation

5.17.1.1 const int anonymous_namespace{matrixgraph_test.h}::MTXNUM =10

Definition at line 25 of file matrixgraph_test.h.

5.18 anonymous_namespace{minqueue_test.h} Namespace Reference

Variables

- const int Q_NUM =10

5.18.1 Variable Documentation

5.18.1.1 const int anonymous_namespace{minqueue_test.h}::Q_NUM =10

Definition at line 26 of file minqueue_test.h.

5.19 anonymous_namespace{prim_test.h} Namespace Reference

Variables

- const int PRIM_N = 10

5.19.1 Variable Documentation

5.19.1.1 const int anonymous_namespace{prim_test.h}::PRIM_N = 10

Definition at line 30 of file prim_test.h.

5.20 anonymous_namespace{relabeltofront_test.h} Namespace Reference

Variables

- const int RTF_N = 6

5.20.1 Variable Documentation

5.20.1.1 const int anonymous_namespace{relabeltofront_test.h}::RTF_N = 6

Definition at line 30 of file relabeltofront_test.h.

5.21 anonymous_namespace{searchtree_test.h} Namespace Reference

Variables

- const int NODE_NUM =9

5.21.1 Variable Documentation

5.21.1.1 const int anonymous_namespace{searchtree_test.h}::NODE_NUM =9

Definition at line 28 of file searchtree_test.h.

5.22 anonymous_namespace{strongconnectedcomponent_test.h} Namespace Reference

Variables

- const int SCC_N = 10

5.22.1 Variable Documentation

5.22.1.1 const int anonymous_namespace{strongconnectedcomponent_test.h}::SCC_N = 10

Definition at line 30 of file strongconnectedcomponent_test.h.

5.23 anonymous_namespace{topologysort_test.h} Namespace Reference

Variables

- const int TPS_N = 10

5.23.1 Variable Documentation

5.23.1.1 const int anonymous_namespace{topologysort_test.h}::TPS_N = 10

Definition at line 31 of file topologysort_test.h.

5.24 IntroductionToAlgorithm Namespace Reference

Namespace of IntroductionToAlgorithm.

Namespaces

- DynamicProgrammingAlgorithm

  Namespace of DynamicProgrammingAlgorithm.
- GraphAlgorithm

  Namespace of GraphAlgorithm.
- QueueAlgorithm

  Namespace of QueueAlgorithm.
- SelectAlgorithm

  Namespace of SelectAlgorithm.
- SetAlgorithm

  Namespace of SetAlgorithm.
- SortAlgorithm

  Namespace of SortAlgorithm.
- StringMatchingAlgorithm

  Namespace of StringMatchingAlgorithm.
- TreeAlgorithm

  Namespace of TreeAlgorithm.

5.24.1 Detailed Description

Namespace of IntroductionToAlgorithm.

5.25 IntroductionToAlgorithm::DynamicProgrammingAlgorithm Namespace Reference

Namespace of DynamicProgrammingAlgorithm.

Functions

- template<typename Iterator , typename OutIterator >
  std::size_t make_LCS (const Iterator begin, const Iterator end, const std::vector< std::vector< int >> &flag↩
  _matrix, typename std::iterator_traits< Iterator >::difference_type seq1_index, typename std::iterator_traits<
  Iterator >::difference_type seq2_index, OutIterator &out_begin)

make_LCS

- template<typename Iterator1 , typename Iterator2 , typename OutIterator >
  std::size_t longest_common_subsequence (const Iterator1 first_begin, const Iterator1 first_end, const Iterator2 second_begin, const Iterator2 second_end, OutIterator out_begin)

  longest_common_subsequence 159.4

5.25.1 Detailed Description

Namespace of DynamicProgrammingAlgorithm.

.

-
-
- >

5.25.2 Function Documentation

5.25.2.1 template<typename Iterator1 , typename Iterator2 , typename OutIterator > std::size_t IntroductionToAlgorithm::DynamicProgrammingAlgorithm::longest_common_subsequence (const Iterator1 *first_begin,* const Iterator1 *first_end,* const Iterator2 *second_begin,* const Iterator2 *second_end,* OutIterator *out_begin*)

longest_common_subsequence 159.4

Parameters

first_begin	:
first_end	
second_begin	:
second_end	
out_begin	

Returns

- X=< x1,x2,...xm > Y=<y1,y2,...yn> Z=<z1,z2,...zk>XY
  - xm=ynzk=xm=yn,Z(k-1)X(m-1)Y(n-1)
  - xm != yn, zk!=xm, Z X(m-1) Y
  - xm != yn, zk!=yn, Z Xm Y(n-1)

  xm=yn, X(m-1)Y(n-1)xm != ynX(m-1) Y,Xm Y(n-1) XY

c[i,j]XiYj c[i,j]= 0 i=0j=0) ;c[i-1,j-1]+1 (i,j>0,xi=yj)max(c[i,j-1],c[i-1,j])(x,j>0 xi!=yj)

- O(m∗n)O(m∗n)

Definition at line 113 of file longest_common_subsequence.h.

5.25.2.2 template< typename Iterator , typename OutIterator > std::size_t IntroductionToAlgorithm::DynamicProgramming← Algorithm::make_LCS (const Iterator *begin,* const Iterator *end,* const std::vector< std::vector< int >> & *flag_matrix,* typename std::iterator_traits< Iterator >::difference_type *seq1_index,* typename std::iterator_traits< Iterator >::difference_type *seq2_index,* OutIterator & *out_begin*)

make_LCS

Parameters

begin	:
end	
flag_matrix	
seq1_index	X[0..seq1_index1]0
seq2_index	Y[0..seq1_index2]0
out_begin	

Returns

- X=< x1,x2,...xm > Y=<y1,y2,...yn> Z=<z1,z2,...zk>XY
  - xm=ynzk=xm=yn,Z(k-1)X(m-1)Y(n-1)
  - xm != yn, zk!=xm, Z X(m-1) Y
  - xm != yn, zk!=yn, Z Xm Y(n-1)

  c[i,j]XiYj
- c[i,j]= 0 i=0j=0)
- c[i,j]=c[i-1,j-1]+1 (i,j>0,xi=yj)
- c[i,j]=max(c[i,j-1],c[i-1,j])(x,j>0 xi!=yj)

flag_matrixc[i-1,j-1]c[i,j-1]c[i-1,j]c[i,j]flag_matrixi,j)

c[i][j] c[i+1][j]
c[i+1][j] c[i+1][j+1]

- xi=yjflag_matrix[i-1][j-1] 11<x1...xi><y1...yj><x1...x(i-1)><y1...y(j-1)>X(i-1)Y(j-1)
- xi=yjc[i-1,j]> c[i,j-1]flag_matrix[i-1][j-1] 10<x1...xi><y1...yj><x1...x(i-1)><y1...yj>X(i-1)Yj
- xi=yjc[i,j-1]> c[i-1,j]flag_matrix[i-1][j-1] 01<x1...xi><y1...yj><x1...x><y1...y(j-1)>XiY(j-1)

O(m+n) O(m∗n)

Definition at line 62 of file longest_common_subsequence.h.

5.26 IntroductionToAlgorithm::GraphAlgorithm Namespace Reference

Namespace of GraphAlgorithm.

Classes

- struct ADJListGraph

  ADJListGraph2222.1
- struct BFS_Vertex

  BFS_Vertex2222.2
- struct DFS_Vertex

  DFS_Vertex2222.3
- struct Edge

  Edge2222.1
- struct FlowVertex

FlowVertex-2626.4

- struct FrontFlowVertex

    FrontFlowVertexrelabel_to_front2626.4

- struct Graph

    Graph2222.1

- struct List

    List

- struct ListNode

    ListNode

- struct MatrixGraph

    MatrixGraph2222.1

- struct SetVertex

    SetVertexnode2222.1

- struct Vertex

    Vertex2222.1

- struct VertexP

    VertexPparent2222.1

Functions

- template< typename GraphType >
  std::pair< std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::↩
  NUM >, std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > > floyd_warshall (std::shared_ptr< GraphType > graph)

    floyd_warshallfloyd_warshall2525.2

- template< typename GraphType >
  std::shared_ptr< Graph< GraphType::NUM+1, typename GraphType::VertexType > > graph_plus_1v (std↩
  ::shared_ptr< GraphType > graph)

    graph_plus_1vgraph2525.2

- template< typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > johnson (std::shared_ptr< GraphType > graph)

    johnsonjohnson2525.3

- template< typename MatrixType >
  MatrixType extend_path (const MatrixType &L, const MatrixType &W)

    extend_path2525.1

- template< typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > matrix_shortest_path (std::shared_ptr< GraphType > graph)

    matrix_shortest_path2525.1

- template< typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > matrix_shortest_path_fast (std::shared_ptr< GraphType > graph)

    matrix_shortest_path2525.1

- template< typename GraphType >
  void connected_component (std::shared_ptr< GraphType > graph)

    connected_component2121.1

- template< typename GraphType >
  bool same_component (std::shared_ptr< GraphType > graph, typename GraphType::VIDType id1, typename GraphType::VIDType id2)

    same_component2121.1

- template<typename GraphType >
  void breadth_first_search (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source↩
  _id, std::function< void(typename GraphType::VIDType)> pre_action=[ ](typename GraphType::VIDType){}, std::function< void(typename GraphType::VIDType)> post_action=[ ](typename GraphType::VIDType){})

  breadth_first_search2222.2
- template<typename GraphType >
  void visit (std::shared_ptr< GraphType > graph, typename GraphType::VIDType v_id, int &time, std↩
  ::function< void(typename GraphType::VIDType, int)> pre_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> post_action=[ ](typename GraphType::VIDType, int){})

  visit2222.3
- template<typename GraphType >
  void depth_first_search (std::shared_ptr< GraphType > graph, std::function< void(typename GraphType::↩
  VIDType, int)> pre_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename Graph↩
  Type::VIDType, int)> post_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> pre_root_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> post_root_action=[ ](typename GraphType::VIDType, int){}, const std::vector< typename GraphType::VIDType > &search_order=std::vector< typename GraphType::VIDType >())

  depth_first_search2222.3
- template<typename GraphType >
  const std::vector< std::vector< typename GraphType::VIDType > > scc (std::shared_ptr< GraphType > graph)

  scc2222.5
- template<typename GraphType >
  std::vector< typename GraphType::VIDType > topology_sort (std::shared_ptr< GraphType > graph)

  topology_sort2222.4
- template<typename GraphType >
  std::shared_ptr< GraphType > create_Gf (const std::shared_ptr< GraphType > graph, std::array< std↩
  ::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  create_Gf2626.2
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > ford↩
  _fulkerson (const std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  ford_fulkersonford_fulkerson2626.2
- template<typename GraphType >
  void push (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, typename Graph↩
  Type::VIDType v_id, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  pushgeneric_push_relabelpush2626.4
- template<typename GraphType >
  GraphType::VIDType min_v_at_Ef (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, const std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType↩
  ::NUM > &flow)

  min_v_at_Efrelabelmin_v_at_Ef2626.4
- template<typename GraphType >
  void relabel (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, const std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  relabelgeneric_push_relabelrelabel2626.4
- template<typename GraphType >
  void initialize_preflow (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  initialize_preflowgeneric_push_relabel2626.4

- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > generic_push_relabel (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  generic_push_relabel-2626.4
- template<typename GraphType >
  void discharge (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, std::array< std↩
  ::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  discharge2626.5
- template<typename GraphType >
  List< ListNode< typename GraphType::VertexType > > create_L (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  create_LL
- template<typename GraphType >
  void initial_vertex_NList (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  initial_vertex_NList
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > relabel_to_front (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  relabel_to_front2626.5
- template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType,typename GraphType↩
  ::VIDType)>>
  GraphType::EWeightType kruskal (std::shared_ptr< GraphType > graph, ActionType pre_action=[ ](typename GraphType::VIDType, typename GraphType::VIDType){}, ActionType post_action=[ ](typename GraphType↩
  ::VIDType, typename GraphType::VIDType){})

  kruskalKruskal2323.2
- template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType)>>
  GraphType::EWeightType prim (std::shared_ptr< GraphType > graph, typename GraphType::VI↩
  DType source_id, ActionType pre_action=[ ](typename GraphType::VIDType){}, ActionType post_↩
  action=[ ](typename GraphType::VIDType){})

  primPrim2323.2
- template<typename GraphType >
  void initialize_single_source (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source↩
  _id)

  initialize_single_source2424.1
- template<typename VertexType >
  void relax (std::shared_ptr< VertexType > from, std::shared_ptr< VertexType > to, typename VertexType↩
  ::KeyType weight)

  relax2424.1
- template<typename GraphType >
  bool bellman_ford (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

  bellman_fordbellman_ford2424.1
- template<typename GraphType >
  void dag_shortest_path (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

  dag_shortest_pathdag_shortest_path2424.2
- template<typename GraphType >
  void dijkstra (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

  dijkstradijkstra2424.3
- template<typename T >
  T unlimit ()

  unlimit
- template<typename T >
  bool is_unlimit (T t)

is_unlimit

- template<typename VertexType >
  std::vector< typename VertexType::VIDType > get_path (const std::shared_ptr< VertexType > v_from, const std::shared_ptr< VertexType > v_to)

  get_path

- template<typename MatrixType >
  std::string matrix_string (const MatrixType &matrix)

  matrix_string

5.26.1 Detailed Description

Namespace of GraphAlgorithm.

5.26.2 Function Documentation

5.26.2.1 template<typename GraphType > bool IntroductionToAlgorithm::GraphAlgorithm::bellman_ford (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *source_id*)

bellman_fordbellman_ford2424.1

Parameters

graph:	
source_↩ id<tt>id</tt>	

Returns

: true

G=(V,E)w:E->Rp=<v0,v1,...vk> w(p)=w(v0,v1)+w(v1,v2)+...+w(v(k-1),vk)uv delt(u,v)

- min{w(p):u–>v(p)}uv
- uv

uvw(p)=delt(u,v)uvp

G=(V,E)vv.paipaiG_pai=(V_pai,E_pai) V_pai={vV:v.pai!=nil}s E_paiV_paipaiE_pai={(v.pai,v)E:vV_pai-{s} }G_pais s

Bellman-Ford

Bellman-FordG=(V,E)w:E->RBellman-Ford bool

Bellman-Fordsvv.key

-
- |V|-1
-

O(VE)

Definition at line 143 of file bellmanford.h.

5.26.2.2 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::breadth_first_search (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *source_id*, std::function< void(typename GraphType::VIDType)> *pre_action* = `[](typename GraphType::VIDType){}`, std::function< void(typename GraphType::VIDType)> *post_action* = `[](typename GraphType::VIDType){}`)

breadth_first_search2222.2

Parameters

graph:	
source_id<tt>id</tt>	
pre_action:<tt>id</tt>	
post_action:<tt>id</tt>	

Returns

:void

`source_id`

- `source_id[0,N)source_id`
- `graphidsource_idsource_id`

G=(V,E)`scolor parentkey`Q

- `key`
- `key`0
- QQ
- Q
  - Q`v`
  - `v`Q
  - `v`

O(E+V)

ssv delt(s,v) sv sv delt(s,v)= sv delt(s,v) s v

G=(V,E)sG G_pai=(V_pai,E_pai) V_pai={ vV: v.parent!=NIL}{s}E_pai={(v.parent,v):v(V_pai-{s})} V_paissE_paiV_pais BFSG_paisvGsv

Definition at line 68 of file bfs.h.

5.26.2.3 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::connected_component (std::shared_ptr< GraphType > *graph*)

connected_component2121.1

Parameters

graph:	

Returns

:void

connected_componentconnected_componentsame_component

connected_component

- v
- (u,v)uv

Definition at line 44 of file connectedcomponent.h.

5.26.2.4 template< typename GraphType > std::shared_ptr<GraphType> IntroductionToAlgorithm::GraphAlgorithm::create←_Gf (const std::shared_ptr< GraphType > *graph,* std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > & *flow*)

create_Gf2626.2

Parameters

graph:	
flow	

Returns

:

G(V,E)fGf(V,Ef)GfGGf

- (u,v)Ef(u,v)<c(u,v)(u,v)Efcf(u,v)=c(u,v)-f(u,v)Gcf
- (u,v)E(v,u)Ef, cf(v,u)=c(u,v)(u,v)(v,u))cf

  f(u,v)(u,v)c(u,v)G(u,v)cf(u,v)Gf(u,v)

c>0c=0f>=0 G(u,v)(v,u)

graph0

O(V+E)

Definition at line 54 of file fordfulkerson.h.

5.26.2.5 template< typename GraphType > List< ListNode< typename GraphType::VertexType> > IntroductionToAlgorithm::GraphAlgorithm::create_L (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *src,* typename GraphType::VIDType *dst*)

create_LL

Parameters

graph:	
src	
dst	

Returns

: L

srcdst

- `id[0,N)`
- `id`

stL

Definition at line 113 of file relabeltofront.h.

5.26.2.6 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::dag_shortest_path (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *source_id*)

dag_shortest_pathdag_shortest_path2424.2

Parameters

graph:	
source_↩ id<tt>id</tt>	

Returns

: void

G=(V,E)w:E->Rp=<v0,v1,...vk> w(p)=w(v0,v1)+w(v1,v2)+...+w(v(k-1),vk)uv delt(u,v)

- min{w(p):u–>v(p)}uv
- uv

uvw(p)=delt(u,v)uvp

G=(V,E)vv.paipaiG_pai=(V_pai,E_pai) V_pai={vV:v.pai!=nil}s E_paiV_paipaiE_pai={(v.pai,v)E:vV_pai-{s} }G_pais s

dag_shortest_path

dag_shortest_path dag_shortest_pathO(V+E)

-
-
-

O(V+E)

Definition at line 71 of file dagshortpath.h.

5.26.2.7 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::depth_first_search (std::shared_ptr< GraphType > *graph*, std::function< void(typename GraphType::VIDType, int)> *pre_action* = `[](typename GraphType::VIDType,int){}`**, std::function< void(typename GraphType::VIDType, int)> *post_action* =** `[](typename GraphType::VIDType,int){}`**, std::function< void(typename GraphType::VIDType, int)> *pre_root_action* =** `[](typename GraphType::VIDType,int){}`**, std::function< void(typename GraphType::VIDType, int)> *post_root_action* =** `[](typename GraphType↩ ::VIDType,int){}`**, const std::vector< typename GraphType::VIDType > & *search_order* =** `std::vector<typename GraphType::VIDType>()` **)**

depth_first_search2222.3

Parameters

graph:	
pre_root_↩ action↩ :<tt>id</tt><tt>time</tt>	
post_root_↩ action↩ :<tt>id</tt><tt>time</tt>	
pre_action↩ :<tt>id</tt><tt>time</tt>	
post_action↩ :<tt>id</tt><tt>time</tt>	
search_order↩ :)<tt>id</tt>	

Returns

:void

vvvv

vdiscover_timevvfinish_timevv vv.discover_timev.discover_timev.finish_timev.finish_time

- (DFS_Vertexcolor)
- time 0
- Vv -v visit

  visit v
  V

O(V+E)

:

- G_pai
- v u vu
- "(u"u"u)"u uv

  - [u.discover_time,u.finish_time][v.discover_time,v.finish_time] uvvu
  - [u.discover_time,u.finish_time][v.discover_time,v.finish_time]uv
  - [v.discover_time,v.finish_time][u.discover_time,u.finish_time]vu

GG_pai

- v(u,v)(u,v)
- (u,v)uv
- uv
-

(u,v)

- v(u,v)
- v(u,v)
- v(u,v)
  - u.discover_time< v.discover_time
  - u.discover_time< v.discover_time

Definition at line 138 of file dfs.h.

5.26.2.8 template<typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::dijkstra (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *source_id*)

dijkstradijkstra2424.3

Parameters

graph:	
source_↩ id<tt>id</tt>	

Returns

: void

G=(V,E)w:E->Rp=<v0,v1,...vk> w(p)=w(v0,v1)+w(v1,v2)+...+w(v(k-1),vk)uv delt(u,v)

- min{w(p):u–>v(p)}uv
- uv

uvw(p)=delt(u,v)uvp

G=(V,E)vv.paipaiG_pai=(V_pai,E_pai) V_pai={vV:v.pai!=nil}s E_paiV_paipaiE_pai={(v.pai,v)E:vV_pai-{s} }G_pais s

Dijkstra

DijkstraDijkstra

Dijkstra S s S V-S u, uSuQkey

- `initialize_single_source`
- SQ
- Q
  - u
  - uS
  - uQQ`decreate_key()`

O(V^2+E)

Definition at line 77 of file dijkstra.h.

5.26.2.9 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::discharge (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *u_id*, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > & *flow*)

discharge2626.5

Parameters

graph:	
u_id	id
flow	

Returns

: void

- `id[0,N)`
- `id`

u, uudischarge(u)

- u.e>0
  - u.currentv
  - vu.Nurelabelu.currentu.N
  - v push (c_f(u,v)>0 u.h=v.h+1)push
  - v push u.currentu.N

Definition at line 53 of file relabeltofront.h.

5.26.2.10 template< typename MatrixType > MatrixType IntroductionToAlgorithm::GraphAlgorithm::extend_path (const MatrixType & *L*, const MatrixType & *W*)

extend_path2525.1

Parameters

L:L	
W	

Returns

: L

matrix_shortest_path $>$ MatrixTypen$*$nMatrixTypen$*$n

- i 0...N-1(N)
  - j 0...N-1(N)
    - newL[i][j]k,k 0...N-1(N) L[i][k]+W[k][j]newL[i][j]
- newL

O(n^3)

Definition at line 48 of file matrix_shortest_path.h.

5.26.2.11 template$<$typename GraphType $>$ std::pair$<$ std::array$<$std::array$<$typename GraphType::EWeightType ,GraphType::NUM$>$,GraphType::NUM$>$, std::array$<$std::array$<$typename GraphType::EWeightType ,GraphType::NUM$>$,GraphType::NUM$>$ $>$ IntroductionToAlgorithm::GraphAlgorithm::floyd_warshall (std::shared_ptr$<$ GraphType $>$ *graph*)

floyd_warshallfloyd_warshall2525.2

Parameters

graph:	

Returns

: n$*$n(d_i_j)n$*$n(p_i_j)std::pair d_i_j ij, p_i_j ijj

G=(V,E)w:E->Ru,vVu v

nG=(V,E)W=(w_i_j) w_i_j =:

- 0:i=j
- (i,j)i!=j(i,j)E
- i!=j(i,j)E

n$*$nD=(d_i_j) d_i_j ij

II=(pai_i_j) pai_i_ji=jij NILijjIIii

floyd_warshall

floyd_warshallGV={1,2,3...n},{1,2,...k}kn i,jVij{1,2,...k}p

- kpp{1,2,3,...k-1}ij{1,2,...k-1} ij{1,2...k)
- kpp i–>k(p1)–>j(p2)p1ik{1,2...k-1} p2kj{1,2,...k-1}

d_i_j<k>ij{1,2...k}k=0ij 0d_i_j<0>=w_i_jd_i_j<k>

- w_i_jk=0
- min(d_i_j<k-1>,d_i_k<k-1>+d_k_j<k-1>k>0

{1,2,...n}D<n>=(d_i_j<n>)

D<k>II II<0>,II<1>...II<k>II<k>=(pai_i_j<k>) pai_i_j<k>ij{1,2,...k}j

k=0ij pai_i_j<0>=:

- null:i=jw_i_j=
- i i!=jw_i_j!=

k>=1kpp{1,2,3,...k-1} pai_i_j<k>=pai_i_j<k-1>; kp, i–>k–>j k!=jpai_i_j<k>=pai_k_j<k-1>k>=1pai_i_j<k>=:

- pai_i_j<k-1>d_i_j<k-1> <= d_i_k<k-1>+dk_j<k-1>
- pai_k_j<k-1>: d_i_j<k-1> > d_i_k<k-1>+dk_j<k-1>

- DP
- k 0..N-1(N)
  - D<k>,P<k> i 0..N-1(N)
  - j 0..N-1(N)
    - d_i_j<k>p_i_j<k>(DD<k-1>PP<k-1>
  - D<k>D,P<k>P
- std::make_pair(D,P)

O(V^3)

Definition at line 109 of file floyd_warshall.h.

5.26.2.12 template<typename GraphType > std::array<std::array<typename GraphType::EWeightType,GraphType::NU↩M>,GraphType::NUM> IntroductionToAlgorithm::GraphAlgorithm::ford_fulkerson (const std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *src*, typename GraphType::VIDType *dst*)

ford_fulkersonford_fulkerson2626.2

Parameters

graph:	
src	
dst	

Returns

:

srcdst

- `id[0,N)`
- `id`

G=V,E(u,v)Ec(u,v)>0E(u,v) (v,u)(u,v)E,c(u,v)=00

ststvE s–>v–>t

Gf:V∗V–>R

- u,vV0<= f(u,v) <= c(u,v)
- uV-{s,t}uu

(u,v)Euvf(u,v)=0f(u,v)uvf |f|=-

Gst,

ford_fulkerson

ford_fulkerson

G(V,E)fGf(V,Ef) GfGGf

- (u,v)Ef(u,v)<c(u,v)(u,v)Efcf(u,v)=c(u,v)-f(u,v)Gcf
- (u,v)E(v,u)Ef, cf(v,u)=c(u,v)(u,v)(v,u))cf

  f(u,v)(u,v)c(u,v)G(u,v)cf(u,v)Gf(u,v)

G=(V,E)fpGfst pp cf(p)= min {cf(u,v):(u,v)p}

G=(V,E)(S,T)VST=V-SsStT(S,T) c(S,T)=c(u,v)uS,vT

fG=(V,E)st

- fG
- Gf
- |f|=c(S,T)(S,T)G

ford_fulkersonu,vVf(u,v)=0 GGGf

- flowflow[i][j]0
-
    - flowGGf
    -
    -
        * cf(p)= min {cf(u,v):(u,v)p}p(u,v)
            - (u,v)GE flow[u][v]=flow[u][v]+cf(p)
            - (v,u)GE flow[u][v]=flow[u][v]-cf(p)
- flow

ford_fulkerson

O(E|f∗|) ,|f∗|

Definition at line 178 of file fordfulkerson.h.

5.26.2.13 template<typename GraphType > std::array<std::array<typename GraphType::EWeightType,GraphType::N↩ UM>,GraphType::NUM> IntroductionToAlgorithm::GraphAlgorithm::generic_push_relabel (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *src,* typename GraphType::VIDType *dst*)

generic_push_relabel-2626.4

Parameters

graph:	
src	
dst	

Returns

:

srcdst

- `id[0,N)`
- `id`

G=V,E(u,v)Ec(u,v)>0E(u,v) (v,u)(u,v)E,c(u,v)=00

ststvE s–>v–>t

Gf:V∗V–>R

- u,vV0<= f(u,v) <= c(u,v)
- uV-{s,t}uu

(u,v)Euvf(u,v)=0f(u,v)uvf |f|=-

Gst,

generic_push_relabel

-- Ford-fulkserson Ford-fulkerson - V∗V −> Rf, u -=e(u)u uV-{s,t} e(u)>0u

G=(V,E)

- e
- h

hh(s)=|V|,h(t)=0,(u,v)E_f h(u)<=h(v)+1

hpush|V|00

generic_push_relabel

uuuuu u1u u

push

u(v,u)f(v,u)>0G_fc_f(u,v)>0h(u)=h(v)+1 (u,v)push

uu.e(u,v)uvmin(u.e,c_f(u,v)) u.e c(u,v)

push

- c_f(u,v)
- delt_f=min(u.e,c_f(u,v))
-
  - (u,v) E f(u,v) += delt_f
  - (v,u) E f(v,u) -= delt_f
-
  - u.e -= delt(u,v)
  - v.e += delt(u,v)

relabel

u(u,v)E_f(G_f)u.h<=v.hurelabel relabelE_fu

relabel

- min{v.h:(u,v)E_f}
- u.h=1+ min{v.h:(u,v)E_f}

-
  - flow: flow(u,v)=c(u,v)u=s; flow(u,v)=0
  - h: h(s)=|V|;h(u)=0, u V-{s}
  - e e(u)=c(s,u)us; e(u)=0us; e(s)s
- uV-{s,t}e
  - push push
  - push e>0 relabel relabel

O(V^2 E)

Definition at line 373 of file genericpushrelabel.h.

5.26.2.14 template< typename VertexType > std::vector< typename VertexType::VIDType> IntroductionToAlgorithm::Graph↩ Algorithm::get_path (const std::shared_ptr< VertexType > *v_from*, const std::shared_ptr< VertexType > *v_to*)

get_path

Parameters

v_from	
v_to	

Returns

: id

`v_fromv_toidstd::vector<typename VertexType::VIDType>`

`v_fromv_to`

Definition at line 141 of file header.h.

5.26.2.15 template< typename GraphType > std::shared_ptr< Graph< GraphType::NUM+1,typename GraphType::VertexType> > IntroductionToAlgorithm::GraphAlgorithm::graph_plus_1v (std::shared_ptr< GraphType > *graph*)

graph_plus_1vgraph2525.2

Parameters

graph:	

Returns

:

graphnew_graphnew_graph graphs new_graphgraph{(s,v):vgraph}; new_graphgraph w(s,v)=0 >NN+1

johnson

Definition at line 46 of file johnson.h.

5.26.2.16 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::initial_vertex_NList (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *src*, typename GraphType::VIDType *dst*)

initial_vertex_NList

Parameters

graph:	
src	

dst	

Returns

: void

srcdst

- `id[0,N)`
- `id`

st

Definition at line 153 of file relabeltofront.h.

5.26.2.17 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::initialize_preflow (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *src*, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > & *flow*)

initialize_preflowgeneric_push_relabel2626.4

Parameters

graph:	
src	
flow	

Returns

: void

src

- `id[0,N)`
- `id`

- flow: flow(u,v)=c(u,v)u=s; flow(u,v)=0
- h: h(s)=|V|;h(u)=0, u V-{s}
- e e(u)=c(s,u)us; e(u)=0us; e(s)s

    `key`e

Definition at line 227 of file genericpushrelabel.h.

5.26.2.18 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::initialize_single_source (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *source_id*)

initialize_single_source2424.1

Parameters

graph:	
source_↩ id<tt>id</tt>	

Returns

: void

`source_id`

- `source_id[0,N)source_id`
- `graphidsource_idsource_id`

`keyparentkey`0

O(V)

Definition at line 43 of file bellmanford.h.

5.26.2.19 template<typename T > bool IntroductionToAlgorithm::GraphAlgorithm::is_unlimit (T *t*)

is_unlimit

Parameters

t	

Returns

: `truefalse`

true;false

`std::numeric_limits<T>::max()/3` >

Definition at line 126 of file header.h.

5.26.2.20 template<typename GraphType > std::array<std::array<typename GraphType::EWeightType ,GraphType::NUM>,GraphType::NUM> IntroductionToAlgorithm::GraphAlgorithm::johnson (std::shared_ptr< GraphType > *graph*)

johnsonjohnson2525.3

Parameters

graph:	

Returns

: n∗n(d_i_j) d_i_j ij

G=(V,E)w:E->Ru,vVu v

nG=(V,E)W=(w_i_j) w_i_j =:

- 0:i=j

- (i,j)i!=j(i,j)E
- i!=j(i,j)E

n∗nD=(d_i_j) d_i_j ij

II=(pai_i_j) pai_i_ji=jij NILijjIIii

Johnson

JohnsonFloyd-WarshallJohnson JohnsonDijkstraBellman-Ford

JohnsonG=(V,E)wDijkstra G w'

- u,vVpwuvpw' uv
- (u,v)w'(u,v)

G=(V,E)wE->R G'=(V',E')V'=V{s}ssV E'=E{(s,v);vV}wvVw(s,v)=0 vV'h(v)=delt(s,v)w'(u,v)=w(u,v)+h(u)-h(v)

-
  - new_graph
  - bellman_ford s
  -
  - hnew_graph
- new_graphsv,vdijkstraD[i][j] new_graph ij
- D

OV^2 lgV + VE)

Definition at line 139 of file johnson.h.

5.26.2.21 template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType,typename GraphType::VIDType)>> GraphType::EWeightType IntroductionTo↩ Algorithm::GraphAlgorithm::kruskal (std::shared_ptr< GraphType > *graph*, ActionType *pre_action* = `[](typename GraphType::VIDType,typename GraphType::VIDType){}`, ActionType *post_action* = `[](typename GraphType::VIDType,typename GraphType::VIDType){}`)

kruskalKruskal2323.2

Parameters

graph:	
source_↩ id<tt>id</tt>	

pre_action↩ :<tt>id</tt>	
post_action↩ :<tt>id</tt>	

Returns

:

G=(V,E)(u,v)Ew(u,v)TTEVT TTTG

A

(u,v)AA(u,v)A

Kruskal

KruskalAGKruskal(u,v) Kruskal

- AGv,
- GE
- E(u,v)
  - uv(u,v)Auv

>AABUG

Kruskal21.3src/set_algorithms/disjoint_set Kruskal O(ElgV)

Definition at line 69 of file kruskal.h.

5.26.2.22 template<typename GraphType > std::array<std::array<typename GraphType::EWeightType ,GraphType::NUM>,GraphType::NUM> IntroductionToAlgorithm::GraphAlgorithm::matrix_shortest_path (std::shared_ptr< GraphType > *graph*)

matrix_shortest_path2525.1

Parameters

graph:	

Returns

: n∗n(d_i_j) d_i_j ij

G=(V,E)w:E->Ru,vVu v

nG=(V,E)W=(w_i_j) w_i_j =:

- 0:i=j
- (i,j)i!=j(i,j)E
- i!=j(i,j)E

n∗nD=(d_i_j) d_i_j ij

II=(pai_i_j) pai_i_ji=jij NILijjIIii

matrix_shortest_path

matrix_shortest_pathijppmm i=jp0ijp i–>k(p')–>jp'm-1

l_i_j<m>ijml_i_j<m>=

- 0i=j
- : i!=j

m>=1 l_i_j<m>=min(l_i_j<m-1>min_(1<=k<=n){l_i_k<m-1>+w_k_j})=min_(1<=k<=n){l_i_k<m-1>+w_k_j} Gi,jdelt(i,j)<ij n-1delt(i,j)=l_i_j<n-1>=l_i_j<n>=...

matrix_shortest_pathW=(w_i_j) L<1>L<2>,...L<n-1>L<n-1> L<1>=Wextend_path

- W
- L L<0>=W, L<k>=extend_path(L<k-1>,W)
- L<N-1>

O(V^4)

Definition at line 129 of file matrix_shortest_path.h.

5.26.2.23 template<typename GraphType > std::array<std::array<typename GraphType::EWeightType ,GraphType::NUM>,GraphType::NUM> IntroductionToAlgorithm::GraphAlgorithm::matrix_shortest_path_fast (std::shared_ptr< GraphType > *graph*)

matrix_shortest_path2525.1

Parameters

graph:	

Returns

: n∗n(d_i_j) d_i_j ij

G=(V,E)w:E->Ru,vVu v

nG=(V,E)W=(w_i_j) w_i_j =:

- 0:i=j
- (i,j)i!=j(i,j)E
- i!=j(i,j)E

n∗nD=(d_i_j) d_i_j ij

II=(pai_i_j) pai_i_ji=jij NILijjIIii

matrix_shortest_path_fast

matrix_shortest_path_fastmatrix_shortest_pathL<m>L<n-1> matrix_shortest_pathtL<n-1> nlg(n-1)

- L<1>=W
- L<2>=W^2=W.W
- L<4>=W^4=W^2.W^2
- L<8>=W^8=W^4.W^4 ....

- W
- L L<0>=W, L<2∗k>=extend_path(L<k>,L<k>)
- L<log(N-1)>

O(V^3lgV)

Definition at line 211 of file matrix_shortest_path.h.

5.26.2.24 template<typename MatrixType > std::string IntroductionToAlgorithm::GraphAlgorithm::matrix_string (const MatrixType & *matrix*)

matrix_string

Parameters

matrix	

Returns

:

`matrix`

Definition at line 169 of file header.h.

5.26.2.25 template<typename GraphType > GraphType::VIDType IntroductionToAlgorithm::GraphAlgorithm::min_v_at_Ef (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *u_id*, const std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > & *flow*)

min_v_at_Efrelabelmin_v_at_Ef2626.4

Parameters

graph:	
u_id	uid
flow	

Returns

: (u,v)E_f(G_f)v

Efu(u,v)v

Definition at line 124 of file genericpushrelabel.h.

5.26.2.26 template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType)>> GraphType::EWeightType IntroductionToAlgorithm::GraphAlgorithm::prim (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *source_id*, ActionType *pre_action* = `[](typename GraphType::VIDType){}`, ActionType *post_action* = `[](typename GraphType::VIDType){}`)

primPrim2323.2

Parameters

graph:	
source_id<tt>id</tt>	
pre_action:<tt>id</tt>	
post_action:<tt>id</tt>	

Returns

:

`source_id`

- `source_id[0,N)source_id`
- `graphidsource_idsource_id`

G=(V,E)(u,v)Ew(u,v)TTEVT TTTG

A

(u,v)AA(u,v)A

Prim

PrimArVAA A

PrimAAkeyQ vv.keyvv.paiv

- key(
- key0
- Q
-
    - u
    - uvvQw(u,v)<v.key(u,v)vAv.pai=u,v.key=w(u,v) >decreate_key

PrimO(VlgV+ElgV)=O(ElgV)(O(E+VlgV)

Definition at line 77 of file prim.h.

5.26.2.27 template< typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::push (std::shared_ptr< GraphType > *graph*, typename GraphType::VIDType *u_id*, typename GraphType::VIDType *v_id*, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > & *flow*)

pushgeneric_push_relabelpush2626.4

Parameters

graph:	
u_id	uid
v_id	vid
flow	

Returns

: void

u_idv_id

- `id[0,N)`
- `id`

u(v,u)f(v,u)>0G_fc_f(u,v)>0h(u)=h(v)+1 (u,v)push

uu.e(u,v)uvmin(u.e,c_f(u,v)) u.e c(u,v)

push

- c_f(u,v)
- delt_f=min(u.e,c_f(u,v))
-
    - (u,v) E f(u,v) += delt_f
    - (v,u) E f(v,u) -= delt_f
-
    - u.e -= delt(u,v)
    - v.e += delt(u,v)

- `key`e $*$
- push(u,v) (u,v)Efc_f(u,v)$>$0
- push(u,v) u.e$>$0

Definition at line 67 of file genericpushrelabel.h.

5.26.2.28 template$<$typename GraphType $>$ void IntroductionToAlgorithm::GraphAlgorithm::relabel (std::shared_ptr$<$ GraphType $>$ *graph,* typename GraphType::VIDType *u_id,* const std::array$<$ std::array$<$ typename GraphType::EWeightType, GraphType::NUM $>$, GraphType::NUM $>$ & *flow*)

relabelgeneric_push_relabelrelabel2626.4

Parameters

graph:	
u_id	uid
flow	

Returns

: void

u(u,v)E_f(G_f)u.h$<$=v.hurelabel relabelE_fu

relabel

- min{v.h:(u,v)E_f}
- u.h=1+ min{v.h:(u,v)E_f}

(u,v)E_f(G_f)u.h$>$v.h

Definition at line 184 of file genericpushrelabel.h.

5.26.2.29 template$<$typename GraphType $>$ std::array$<$std::array$<$typename GraphType::EWeightType,GraphType::NU↩ M$>$,GraphType::NUM$>$ IntroductionToAlgorithm::GraphAlgorithm::relabel_to_front (std::shared_ptr$<$ GraphType $>$ *graph,* typename GraphType::VIDType *src,* typename GraphType::VIDType *dst*)

relabel_to_front2626.5

Parameters

graph:	
src	
dst	

Returns

:

srcdst

- `id[0,N)`
- `id`

G=V,E(u,v)Ec(u,v)>0E(u,v) (v,u)(u,v)E,c(u,v)=00

ststvE s–>v–>t

Gf:V∗V–>R

- u,vV0<= f(u,v) <= c(u,v)
- uV-{s,t}uu

(u,v)Euvf(u,v)=0f(u,v)uvf |f|=-

Gst,

relabel_to_front

$>$ generic_push_relabel

-

u uu

G=(V,E)stfGh (u,v)c_f(u,v)>0h(u)=h(v)+1(u,v)(u,v) G_f_h=(V,E_f_h)E_f_h

G=(V,E)uVu.NuG u(u,v)E(v,u)Evu.N >u.N(u,v)v

u.N.headu.Nv.next-neighboru.Nvv NIL

uu.currentu.N u.currentu.N.head

discharge

u, uudischarge(u)

- u.e>0
  - u.currentv
  - vu.Nurelabelu.currentu.N
  - v push (c_f(u,v)>0 u.h=v.h+1)push
  - v push u.currentu.N

- generic_push_relabel
- stL
- stu,u.currentu.N.head
- uL.head
- u!=NIL
  - u.holdh
  - udischarge
  - u.h>oldhuuL
  - u=u.nextuL

O(V^3)

Definition at line 280 of file relabeltofront.h.

5.26.2.30 template<typename VertexType > void IntroductionToAlgorithm::GraphAlgorithm::relax (std::shared_ptr< VertexType > *from,* std::shared_ptr< VertexType > *to,* typename VertexType::KeyType *weight*)

relax2424.1

Parameters

from:	
tofromfrom	
weight:	

Returns

: void

vv.keysvv.keysv

sv su(u,v)svv.keyv.parent

O(1)

Definition at line 82 of file bellmanford.h.

5.26.2.31 template<typename GraphType > bool IntroductionToAlgorithm::GraphAlgorithm::same_component (std::shared_ptr< GraphType > *graph,* typename GraphType::VIDType *id1,* typename GraphType::VIDType *id2*)

same_component2121.1

Parameters

graph:	
id1:	
id2:	id

- id0`GraphType::NUM`
- `graph->vertexes.at(id1)`

same_component connected_component

Definition at line 93 of file connectedcomponent.h.

5.26.2.32 template<typename GraphType > const std::vector<std::vector<typename GraphType::VIDType> > IntroductionToAlgorithm::GraphAlgorithm::scc (std::shared_ptr< GraphType > *graph*)

scc2222.5

Parameters

graph:	

Returns

`:std::vectoridstd::vector`

G=(V,E)CCVCu,vu–>vv–>uuv

GG_TG_T=(V,E_T),E_T={(u,v):(v,u)E}G_TG

- GG_T
- `scc`

- G finish_time
- G_T Gfinish_time
- G_T

O(V+E)

Definition at line 48 of file strongconnectedcomponent.h.

5.26.2.33 template<typename GraphType > std::vector<typename GraphType::VIDType> IntroductionToAlgorithm::GraphAlgorithm::topology_sort (std::shared_ptr< GraphType > *graph*)

topology_sort2222.4

Parameters

graph:	

Returns

`:idstd::vector`

G=VE)G G(u,v)uv

G

G=(V,E)

O(V+E)

Definition at line 45 of file topologysort.h.

5.26.2.34 template<typename T > T IntroductionToAlgorithm::GraphAlgorithm::unlimit ( )

unlimit

Returns

:

:

- `key`
-

`std::numeric_limits<T>::max()/2`

Definition at line 111 of file header.h.

5.26.2.35 template<typename GraphType > void IntroductionToAlgorithm::GraphAlgorithm::visit (std::shared_ptr< Graph↩ Type > *graph*, typename GraphType::VIDType *v_id*, int & *time*, std::function< void(typename GraphType::VIDType, int)> *pre_action* = `[](typename GraphType::VIDType,int){}`, std::function< void(typename GraphType::VIDType, int)> *post_action* = `[](typename GraphType::VIDType,int){}`)

visit2222.3

Parameters

graph:	
v_id↩ :<tt>id</tt>	
time↩ :<tt>visit</tt>	
pre_action↩ :<tt>id</tt><tt>time</tt>	
post_action↩ :<tt>id</tt><tt>time</tt>	`v_id`

- `v_id[0,N)v_id`
- `graphidv_idv_id`

visitv_id

- time
- v_id
- v_id
- v_id time v_id

Definition at line 49 of file dfs.h.

5.27 IntroductionToAlgorithm::QueueAlgorithm Namespace Reference

Namespace of QueueAlgorithm.

Classes

- class MinQueue

  MinQueue66.5

5.27.1 Detailed Description

Namespace of QueueAlgorithm.

5.28 IntroductionToAlgorithm::SelectAlgorithm Namespace Reference

Namespace of SelectAlgorithm.

Functions

- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  std::iterator_traits< Iterator >::value_type good_select (const Iterator begin, const Iterator end, typename std::iterator_traits< Iterator >::difference_type rank, CompareType compare=CompareType())

  good_select 99.3 O(n)
- template<typename IntType >
  IntType radom_index (IntType begin, IntType end)

  radom_index
- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  std::iterator_traits< Iterator >::value_type randomized_select (const Iterator begin, const Iterator end, typename std::iterator_traits< Iterator >::difference_type rank, CompareType compare=CompareType())

  randomized_select 99.2

5.28.1 Detailed Description

Namespace of SelectAlgorithm.

5.28.2 Function Documentation

5.28.2.1 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_↩ type>> std::iterator_traits<Iterator>::value_type IntroductionToAlgorithm::SelectAlgorithm::good_select (const Iterator *begin*, const Iterator *end*, typename std::iterator_traits< Iterator >::difference_type *rank*, CompareType *compare* = `CompareType()`)

good_select 99.3 O(n)

Parameters

begin	:
end	
rank	01....nn
compare	std::less<T>

Returns

rank

- A[p...r]k
  -
    - 515
    -
    -
    - good_select
    -
  - m
  -
    - m==k
    - m<k A[m+1...r](k-m-1)good_select(....)
    - m>k A[p...m-1] k good_select(...)
- O(n)

- good_select(...)

Definition at line 52 of file goodselect.h.

5.28.2.2 template<typename IntType > IntType IntroductionToAlgorithm::SelectAlgorithm::radom_index (IntType *begin*, IntType *end*)

radom_index

Parameters

begin	[begin,end]
end	[begin,end]

Returns

[begin,end]

Definition at line 18 of file randomizedselect.h.

5.28.2.3 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_↩ type>> std::iterator_traits<Iterator>::value_type IntroductionToAlgorithm::SelectAlgorithm::randomized_select (const Iterator *begin*, const Iterator *end*, typename std::iterator_traits< Iterator >::difference_type *rank*, CompareType *compare* = `CompareType()`)

randomized_select 99.2

Parameters

begin	:
end	
rank	01....nn
compare	std::less<T>

Returns

rank

- A[p...r]k
  - q,A[q]
  - A[q]A[q] m
  -
    - m==k A[q]
    - m<k A[q+1...r](k-m-1)randomized_select(q+1,end,k-m-1)
    - m>k A[p...q-1] k randomized_select(begin,q,k)
- O(n^2)O(n)
-

Definition at line 47 of file randomizedselect.h.

5.29 IntroductionToAlgorithm::SetAlgorithm Namespace Reference

Namespace of SetAlgorithm.

Classes

- struct DisjointSetNode

  DisjointSetNode2121.3

5.29.1 Detailed Description

Namespace of SetAlgorithm.

`Set`

5.30 IntroductionToAlgorithm::SortAlgorithm Namespace Reference

Namespace of SortAlgorithm.

Classes

- class Sort_Heap

  Sort_Heap6

Functions

- template< typename Iterator >
  void bucket_sort (const Iterator begin, const Iterator end, const typename std::iterator_traits< Iterator >←
  ::value_type &min_val, const typename std::iterator_traits< Iterator >::value_type &max_val)

  bucket_sort8 8.4
- template< typename Iterator >
  void count_sort (const Iterator begin, const Iterator end, const typename std::iterator_traits< Iterator >←
  ::value_type &max_val)

  count_sort8 8.2
- template< typename Iterator , typename CompareType = std::less< typename std::iterator_traits< Iterator >::value_type>>
  void insert_sort (const Iterator begin, const Iterator end, CompareType compare=CompareType())

  insert_sort 2.1
- template< typename Iterator , typename CompareType = std::less< typename std::iterator_traits< Iterator >::value_type>>
  void merge (const Iterator begin, const Iterator end, const Iterator middle, CompareType compare=Compare←
  Type())

  merge 2.3.1
- template< typename Iterator , typename CompareType = std::less< typename std::iterator_traits< Iterator >::value_type>>
  void merge_sort (const Iterator begin, const Iterator end, CompareType compare=CompareType())

  merge_sort 2.3.1
- template< typename Iterator , typename CompareType = std::less< typename std::iterator_traits< Iterator >::value_type>>
  Iterator partition (const Iterator begin, const Iterator end, const Iterator partition_iter, CompareType compare=CompareType())

  partition 7
- template< typename Iterator , typename CompareType = std::less< typename std::iterator_traits< Iterator >::value_type>>
  void quick_sort (const Iterator begin, const Iterator end, CompareType compare=CompareType())

  quick_sort 7
- template< typename T >
  T digi_on_N (T num, std::size_t n)

  digi_on_N
- template< typename Iterator >
  void radix_sort (const Iterator begin, const Iterator end, std::size_t radix_width)

  radix_sort8 8.3

5.30.1 Detailed Description

Namespace of SortAlgorithm.

5.30.2 Function Documentation

5.30.2.1 template< typename Iterator > void IntroductionToAlgorithm::SortAlgorithm::bucket_sort (const Iterator *begin*, const Iterator *end*, const typename std::iterator_traits< Iterator >::value_type & *min_val*, const typename std::iterator_traits< Iterator >::value_type & *max_val*)

bucket_sort8 8.4

Parameters

begin	:
end	
min_val:	
max_val:	

Returns

void

- A[p...r]hashhash
  - hash a<b hash(a)<hash(b)
  - hash
- O(n)
-

Definition at line 43 of file bucketsort.h.

5.30.2.2 template< typename Iterator > void IntroductionToAlgorithm::SortAlgorithm::count_sort (const Iterator *begin*, const Iterator *end*, const typename std::iterator_traits< Iterator >::value_type & *max_val*)

count_sort8 8.2

Parameters

begin	:
end	
max_val:	

Returns

void

- A[p...r]max_val
  - AA[i]CounterArray[A[i]]
  - CounterArrayA[i]
  - A[i]
- O(n)
- O(n)

>static_assert(...,...)

Definition at line 45 of file countsort.h.

5.30.2.3 template<typename T > T IntroductionToAlgorithm::SortAlgorithm::digi_on_N (T *num,* std::size_t *n*)

digi_on_N

Parameters

num	:
n	01...

Returns

T static_assert(std::is_integral<T>::value,"..."")

Definition at line 39 of file radixsort.h.

5.30.2.4 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> void IntroductionToAlgorithm::SortAlgorithm::insert_sort (const Iterator *begin,* const Iterator *end,* CompareType *compare =* `CompareType()`)

insert_sort 2.1

Parameters

begin	:
end	
compare	std::less<T>

Returns

void

- A[p...r]
  - A[q]A[p...q-1]A[q]A[p...q-1]
- O(n^2)
-

Definition at line 38 of file insertsort.h.

5.30.2.5 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> void IntroductionToAlgorithm::SortAlgorithm::merge (const Iterator *begin,* const Iterator *end,* const Iterator *middle,* CompareType *compare =* `CompareType()`)

merge 2.3.1

Parameters

begin	: begin...middle

end	middle...end
middle	begin...middle
compare	std::less<T>

Returns

void

- A[p...q...r]
  - A[p...q]LA[q...r]R
  - LRA
- O(n)
- O(n)

Definition at line 42 of file mergesort.h.

5.30.2.6 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> void IntroductionToAlgorithm::SortAlgorithm::merge_sort (const Iterator *begin*, const Iterator *end*, CompareType *compare* = `CompareType()`)

merge_sort 2.3.1

Parameters

begin	:
end	
compare	std::less<T>

Returns

void

- A[p...r]
  - A[p...r]2A[p...q-1]A[q...r]1
  - A[p...q-1]A[q...r]
- O(nlgn)
- O(n)

Definition at line 86 of file mergesort.h.

5.30.2.7 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> Iterator IntroductionToAlgorithm::SortAlgorithm::partition (const Iterator *begin*, const Iterator *end*, const Iterator *partition_iter*, CompareType *compare* = `CompareType()`)

partition 7

Parameters

begin	:
end	
partition_iter	
compare	std::less<T>

Returns

:

- A[p...r]A[q]
  - A[q]A[r]A[r]
  - A[p...smaller_next-1]A[r]A[smaller_next...current-1]A[r]A[current]
    - A[current]<A[r] A[current] A[smaller_next], currentsmaller_next
    - A[current]>=A[r], current
- O(n)
-

Definition at line 43 of file quicksort.h.

5.30.2.8 template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_↩ type>> void IntroductionToAlgorithm::SortAlgorithm::quick_sort (const Iterator *begin*, const Iterator *end*, CompareType *compare* = `CompareType()`)

quick_sort 7

Parameters

begin	:
end	
compare	std::less<T>

Returns

void

- A[p...r]
  - A[p...r]A[p...q-1]A[q+1...r]A[q]partition
  - A[p...q-1]A[q+1...r]
- O(n^2) O(nlgn)
-

Definition at line 81 of file quicksort.h.

5.30.2.9 template<typename Iterator > void IntroductionToAlgorithm::SortAlgorithm::radix_sort (const Iterator *begin*, const Iterator *end*, std::size_t *radix_width*)

radix_sort8 8.3

Parameters

begin	:
end	
radix_width	0assert(radix_width!=0)

Returns

void

- A[p...r]RADIXWITHRADIXWITH
  - A
  - A
  - A
- O(d(n+k))d(dk0123...9
-
  -
  -

>static_assert(...,...)

Definition at line 67 of file radixsort.h.

5.31 IntroductionToAlgorithm::StringMatchingAlgorithm Namespace Reference

Namespace of StringMatchingAlgorithm.

Functions

- template<typename Iterator >
  std::iterator_traits< Iterator >::difference_type index_of_M (Iterator beginM, Iterator endM, typename std↩ ::iterator_traits< Iterator >::value_type a)

  index_of_M a3232.3
- template<typename Iterator >
  bool is_end_with (Iterator begin, Iterator k_iter, Iterator q_iter, typename std::iterator_traits< Iterator >↩ ::value_type a)

  is_end_with Pk(Pq a)3232.3
- template<typename PIterator , typename MIterator >
  void get_delta (const PIterator P_begin, const PIterator P_end, const MIterator M_begin, const MIterator M_end, std::vector< std::vector< int >> &delta)

  get_delt 3232.3
- template<typename IteratorT , typename IteratorP , typename IteratorM >
  std::vector< int > finite_automaton_match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end, const IteratorM iterM_begin, const IteratorM iterM_end)

  finite_automaton_match 3232.3
- template<typename IteratorP >
  std::vector< int > get_pai (const IteratorP iterP_begin, const IteratorP iterP_end)

  get_pai KMP3232.4

- template<typename IteratorT , typename IteratorP >
  std::vector< int > kmp_match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP←↩ _begin, const IteratorP iterP_end)

  kmp_match KMP3232.4
- template<typename T >
  T get_h (T radix_d, T len_m, T mod_q)

  get_h rabin_karp get_h 3232.2
- template<typename IteratorT , typename IteratorP >
  std::vector< int > rabin_karp_match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end, unsigned radix_d, unsigned mod_q)

  rabin_karp_match rabin_karp3232.2
- template<typename IteratorT , typename IteratorP >
  std::vector< int > match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end)

  match 32,32.1

5.31.1 Detailed Description

Namespace of StringMatchingAlgorithm.

5.31.2 Function Documentation

5.31.2.1 template<typename IteratorT , typename IteratorP , typename IteratorM > std::vector<int> IntroductionToAlgorithm::StringMatchingAlgorithm::finite_automaton_match (const IteratorT *iterT_begin,* const IteratorT *iterT_end,* const IteratorP *iterP_begin,* const IteratorP *iterP_end,* const IteratorM *iterM_begin,* const IteratorM *iterM_end*)

finite_automaton_match 3232.3

Parameters

iterT_begin	: T
iterT_end	T
iterP_begin	: P
iterP_end	P
iterM_begin	:
iterM_end	

Returns

: std::vector

n T[1...n]mP[1...m] m<=nPTMM={0,1}M={a,b,c,...z} PT

AM 5(Q,q_0,A,M,delt)

- Q:
- q_0Q
- AQ
- M
- delt: Q∗M–> Q

q_0qaq delt(q,a) qA

phai, M∗ M 0e,eM∗. phai M∗Q pai(w)w w M∗ phai(w)A w phai:

- phai(e)=q_0
- phai(wa)=delt(phai(w),a), wM∗,aM

P sigmaP sigma M∗{0,1,....m}

- sigma(x)=max{k:P_kx}sigma(x)xP

P0=esigma(e)=0mPsigma(x)=mPx

P[1...m]

- Q{0,1,...m}q_00m
- qa delt delt(q,a)=sigma(Pq a)

TTT[i]PPjPjTi q=phai(Ti)TiqdeltqPTi q PqTiq=sigma(Ti)

T | 1 | 2 | 3 |.....|i-q+1|...........| i |..............| n | :Ti=T[1...i]

|<----q--->|

P | 1 | 2 |.......| q |....| m | :Pq=P[1...q]

delt)

- Pq0m (q=0P_0=):
  - aaPk (Pq a) k delt(q,a)=k
- delt
- Ti1n:
  - q=delt(q,T[i]) q==m i-m i-m std::vector

>n>=0m>=0()

>TP

O(m^3 |M|)|M| O(n)

Definition at line 250 of file finiteautomatonmatch.h.

5.31.2.2 template< typename PIterator , typename MIterator > void IntroductionToAlgorithm::StringMatchingAlgorithm::get_↩ delta (const PIterator *P_begin,* const PIterator *P_end,* const MIterator *M_begin,* const MIterator *M_end,* std::vector< std::vector< int >> & *delta*)

get_delt 3232.3

Parameters

P_begin	: P
P_end	P
M_begin	:
M_end:	
delta	

Returns

: void

- Pq0m (q=0P_0=):
  - aaPk (Pq a) k delt(q,a)=k

  n>=0m>=0()

  MP

Definition at line 120 of file finiteautomatonmatch.h.

5.31.2.3 template<typename T > T IntroductionToAlgorithm::StringMatchingAlgorithm::get_h (T *radix_d*, T *len_m*, T *mod_q*)

get_h rabin_karp get_h 3232.2

Parameters

radix_d	:
len_m	m
mod_q	:

Returns

: radix_d(len_m-1)mod_q

`radix_qlen_mmod_q`0.0

Definition at line 36 of file rabinkarpmatch.h.

5.31.2.4 template<typename IteratorP > std::vector<int> IntroductionToAlgorithm::StringMatchingAlgorithm::get_pai (const IteratorP *iterP_begin*, const IteratorP *iterP_end*)

get_pai KMP3232.4

Parameters

iterP_begin	: P
iterP_end	P

Returns

: pai

- pai[1]=0,k=0

- q 2 m:PkPmm2
  - k>0 P[k+1]!=P[q] k=pai[k]P[k+1]=P[q]PkPm
  - P[k+1]==P[q]k=k+1pai[q]=k
- pai

  >m>0

Definition at line 42 of file kmp.h.

5.31.2.5 template<typename Iterator > std::iterator_traits<Iterator>::difference_type IntroductionToAlgorithm::StringMatchingAlgorithm::index_of_M (Iterator *beginM,* Iterator *endM,* typename std::iterator_traits< Iterator >::value_type *a*)

index_of_M a3232.3

Parameters

beginM	: M
k_iter	M
a	a

Returns

: a

aM

aMaM

Definition at line 40 of file finiteautomatonmatch.h.

5.31.2.6 template<typename Iterator > bool IntroductionToAlgorithm::StringMatchingAlgorithm::is_end_with (Iterator *begin,* Iterator *k_iter,* Iterator *q_iter,* typename std::iterator_traits< Iterator >::value_type *a*)

is_end_with Pk(Pq a)3232.3

Parameters

begin	: P
k_iter	Pk
q_iter	: Pq
a	a

Returns

: Pk(Pq a)

Pk(Pq a)

k_iter>=begin,q_iter>=begin

Definition at line 83 of file finiteautomatonmatch.h.

5.31.2.7 template<typename IteratorT , typename IteratorP > std::vector<int> IntroductionToAlgorithm::StringMatchingAlgorithm::kmp_match (const IteratorT *iterT_begin,* const IteratorT *iterT_end,* const IteratorP *iterP_begin,* const IteratorP *iterP_end*)

kmp_match KMP3232.4

Parameters

iterT_begin	: T
iterT_end	T
iterP_begin	: P
iterP_end	P

Returns

: std::vector

n T[1...n]mP[1...m] m<=nPTMM={0,1}M={a,b,c,...z} PT

KMP

pai P[1...q]T[s+1,...s+q] s's'>sk<qP[1...k]=T[s'+1,...s'+k]s'>ss'+k=s+q?

T | 1 | 2 | 3 |.....|s+1|............| s+q |..............| n | T[s+q]

|<----q---->|

P | 1 | 2 |.......| q |....| m | :Pq=P[1...q]

T | 1 | 2 | 3 |.....|s+1|..|s'+1|..........| s+q |..............| n | T[s+q]

|<----k---->|

P | 1 | 2 |........| k |..|q|..| m | :Pk=P[1...k]

PqT[s+q]Pq Pk T[s+q]P q-k s s'=s+(q-k)

PkT[s+q]Pq PkPqk<qs'=s+(q-k) k(q-k)

P pai:{1,2,...,m}-> {0,1,2,...,m-1} pai[q]=max{k:k<q PkPq}pai[q]PqP

kmp

KMP paiO(m)paipai[1...m] pai

paiKMPTpai m

pai)

- pai[1]=0,k=0
- q 2 m:PkPmm2
  - k>0 P[k+1]!=P[q] k=pai[k]P[k+1]=P[q]PkPm
  - P[k+1]==P[q]k=k+1pai[q]=k
- pai

- q=0
- i1n:
  - q>0 P[q+1]!=T[i] q=pai[q]
  - P[q+1]==T[i] q=q+1
  - q==mstd::vector q=pai[q](P[q+1])
- std::vector

>n>=0m>=0()

>TP

O(m)O(n) O(n)

Definition at line 148 of file kmp.h.

5.31.2.8 template<typename IteratorT , typename IteratorP > std::vector<int> IntroductionToAlgorithm::StringMatching↩ Algorithm::match (const IteratorT *iterT_begin,* const IteratorT *iterT_end,* const IteratorP *iterP_begin,* const IteratorP *iterP_end*)

match 32,32.1

Parameters

iterT_begin	: T
iterT_end	T
iterP_begin	: P
iterP_end	P

Returns

: std::vector

n T[1...n]mP[1...m] m<=nPTMM={0,1}M={a,b,c,...z} PT

n-m+1s P[1...m]=T[s+1,...s+m]

- T 0∼n-m
- s, T[s+1,s+2,...s+m]P[1...m]
- std::vector

>n>=0m>=0()

>TP

O(m∗n)

Definition at line 47 of file match.h.

5.31.2.9 template<typename IteratorT , typename IteratorP > std::vector<int> IntroductionToAlgorithm::StringMatching←↩ Algorithm::rabin_karp_match (const IteratorT *iterT_begin,* const IteratorT *iterT_end,* const IteratorP *iterP_begin,* const IteratorP *iterP_end,* unsigned *radix_d,* unsigned *mod_q*)

rabin_karp_match rabin_karp3232.2

Parameters

iterT_begin	: T
iterT_end	T
iterP_begin	: P
iterP_end	P
radix_d	M
mod_q	

Returns

: std::vector

n T[1...n]mP[1...m] m<=nPTMM={0,1}M={a,b,c,...z} PT

rabin_karp

M={0,1,2,3...,9}k k

P[1...m]pT[1...n]t_s mT[s+1,...s+m]s=0,1,...,n-m T[s+1,...s+m]=P[1...m]p=t_sO(m)p O(n-m+1)t_spt_sO(m)+O(n-m+1)=O(n)

dd=|M|M

O(m)pt_0:

- p=P[m]+10(P[m-1]+10(P[m-2]+...+10(P[2]+10P[1])...))
- t_0=T[m]+10(T[m-1]+10(T[m-2]+...+10(T[2]+10T[1])...))

O(n-m)t_1t_2...t_<n-m>: t_<s+1>=10(t_s-10^(m-1)∗T[s+1])+T[s+m+1]

pt_sPmp(m) qpt_s O(m)qpO(n-m+1)qt_s h = d^(m-1)(mod q) t_<s+1>=(d(t_s-T[s+1]h)+T[s+m+1]) mod q

q t_s = p (mod q) t_s=ps P[1...m]=T[s+1,...s+m]

- p t_0
- s0n-m(n-m)
  - p=t_ssP[1...m]=T[s+1,...s+m]sstd::vector

- std::vector

>n>=0m>0

>TP

rabin_karp O(m)O((n-m+1)m)

Definition at line 114 of file rabinkarpmatch.h.

5.32 IntroductionToAlgorithm::TreeAlgorithm Namespace Reference

Namespace of TreeAlgorithm.

Classes

- struct BinaryTree

  BinaryTree1010.4
- struct BinaryTreeNode

  BinaryTreeNodexxxx
- class SearchTree

  SearchTree12

Functions

- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void inorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  inorder_walk
- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void preorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  preorder_walk
- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void postorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  postorder_walk
- template<typename NodeType >
  void left_rotate (std::shared_ptr< NodeType > node, std::shared_ptr< NodeType > &root)

  left_rotate
- template<typename NodeType >
  void right_rotate (std::shared_ptr< NodeType > node, std::shared_ptr< NodeType > &root)

  right_rotate
- template<typename NodeType >
  void transplant (std::shared_ptr< NodeType > node_src, std::shared_ptr< NodeType >node_dst, std↵ ::shared_ptr< NodeType > &root)

  transplant

5.32.1 Detailed Description

Namespace of TreeAlgorithm.

5.32.2 Function Documentation

5.32.2.1 template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>> void IntroductionToAlgorithm::TreeAlgorithm::inorder_walk (std::shared_ptr< NodeType > *root*, ActionType *action* = `[](typename NodeType::T){}`)

inorder_walk

Parameters

root	
func	

Returns

void

-
-
-

O(n)O(1)

Definition at line 77 of file binarytree.h.

5.32.2.2 template<typename NodeType > void IntroductionToAlgorithm::TreeAlgorithm::left_rotate (std::shared_ptr< NodeType > *node*, std::shared_ptr< NodeType > & *root*)

left_rotate

Parameters

node	
root	noderoot

Returns

void

nodel_noder_node r_nodenodenoder_noder_nodenode

```
       |                                              |
      node                                          r_node
     /    \                                         /    \
 l_node   r_node        --   -->               node   r_r_node
         /     \                                  /     \
   l_r_node r_r_node                          l_node  l_r_node
```

O(1)O(1)

Definition at line 157 of file binarytree.h.

5.32.2.3 template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>> void IntroductionToAlgorithm::TreeAlgorithm::postorder_walk (std::shared_ptr< NodeType > *root*, ActionType *action* = `[](typename NodeType::T){}`)

postorder_walk

Parameters

root	
func	

Returns

void

-
-
-

O(n)O(1)

Definition at line 127 of file binarytree.h.

5.32.2.4 template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>> void IntroductionToAlgorithm::TreeAlgorithm::preorder_walk (std::shared_ptr< NodeType > *root*, ActionType *action* = `[](typename NodeType::T){}`)

preorder_walk

Parameters

root	
func	

Returns

void

-
-
-

O(n)O(1)

Definition at line 101 of file binarytree.h.

5.32.2.5 template<typename NodeType > void IntroductionToAlgorithm::TreeAlgorithm::right_rotate (std::shared_ptr< NodeType > *node*, std::shared_ptr< NodeType > & *root*)

right_rotate

Parameters

node	

root	noderoot

Returns

void

nodel_noder_node l_nodenodenoder_nodel_nodenode

```
                  |                                                |
                 node                                            l_node
               /     \                                          /    \
           l_node   r_node         --   -->             l_l_node   node
          /     \                                                      /    \
     l_l_node r_l_node                                          r_l_node   r_node
```

O(1)O(1)

Definition at line 204 of file binarytree.h.

5.32.2.6 template< typename NodeType > void IntroductionToAlgorithm::TreeAlgorithm::transplant (std::shared_ptr< NodeType > *node_src*, std::shared_ptr< NodeType > *node_dst*, std::shared_ptr< NodeType > & *root*)

transplant

Parameters

node_src	
root	noderoot
node_dst	

Returns

void

node_srcnode_dst–>–>

```
                src_p         dst_p                                     src_p         dst_p
                  ||            ||                                 (-->)|              ||
                node_src     node_dst           --- -->              node_src      node_src
                //   \       //    \                                 //     \      //     \
```

O(1)O(1)

Definition at line 249 of file binarytree.h.

Chapter 6

Class Documentation

6.1 IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N > Struct Template Reference

ADJListGraph2222.1

`#include <adjlistgraph.h>`

Public Types

- typedef int VIDType
- typedef int EWeightType
- typedef std::tuple< VIDType, VIDType, EWeightType > EdgeTupleType

Public Member Functions

- void add_edge (const EdgeTupleType &edge_tuple)

  add_edge:
- template<typename Iteator >
  void add_edges (const Iteator &begin, const Iteator &end)

  add_edges:
- void adjust_edge (VIDType id1, VIDType id2, EWeightType wt)

  adjust_edge:
- const std::vector< EdgeTupleType > edge_tuples () const

  edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`
- const std::vector< EdgeTupleType > vertex_edge_tuples (VIDType id) const

  vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`
- bool has_edge (VIDType id_from, VIDType id_to) const

  has_edge:
- EWeightType weight (VIDType id_from, VIDType id_to) const

  weight:

Public Attributes

- std::array< std::vector< std::pair< VIDType, EWeightType > >, N > array

Static Public Attributes

- static const unsigned NUM =N

6.1.1 Detailed Description

template<unsigned N>struct IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >

ADJListGraph2222.1

- `arraystd::array<std::vector<std::pair<VIDType,EWeightType>>,N>N`

Definition at line 35 of file adjlistgraph.h.

6.1.2 Member Typedef Documentation

6.1.2.1 template<unsigned N> typedef std::tuple<VIDType,VIDType,EWeightType> IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::EdgeTupleType

12)

Definition at line 39 of file adjlistgraph.h.

6.1.2.2 template<unsigned N> typedef int IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::EWeightType

Definition at line 38 of file adjlistgraph.h.

6.1.2.3 template<unsigned N> typedef int IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::VIDType

Definition at line 37 of file adjlistgraph.h.

6.1.3 Member Function Documentation

6.1.3.1 template<unsigned N> void IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::add_edge (const EdgeTupleType & *edge_tuple*) [inline]

add_edge:

Parameters

edge_tuple:	`Edgestd::tuple<VIDType,VIDType,EWeightType>`

`std::invalid_argument`

`[0,N)id`

Definition at line 50 of file adjlistgraph.h.

6.1.3.2 template<unsigned N> template<typename Iteator > void IntroductionToAlgorithm::Graph↩ Algorithm::ADJListGraph< N >::add_edges (const Iteator & *begin*, const Iteator & *end*) [inline]

add_edges:

Parameters

begin:	
end:	Edge`std::tuple<VIDType,VIDType,EWeightType>`

`std::invalid_argument`

`[0,N)id`

Definition at line 71 of file adjlistgraph.h.

6.1.3.3 template<unsigned N> void IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::adjust_edge (VIDType *id1*, VIDType *id2*, EWeightType *wt*) [inline]

adjust_edge:

Parameters

id1:	
id2:	
wt:	`id1id2wtstd::invalid_argument` `id1id2[0,N)id`

Definition at line 92 of file adjlistgraph.h.

6.1.3.4 template<unsigned N> const std::vector<EdgeTupleType> IntroductionToAlgorithm::GraphAlgorithm↩ ::ADJListGraph< N >::edge_tuples ( ) const [inline]

edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Returns

:

Definition at line 113 of file adjlistgraph.h.

6.1.3.5 template<unsigned N> bool IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::has_edge (VIDType *id_from*, VIDType *id_to*) const [inline]

has_edge:

Parameters

id_from	`id`
id_to	`id`

Returns

:

- `id_fromid_to >id_fromid_to[0,N)`
- `id_fromid_totrue`
- `id_fromid_tofalse`

Definition at line 154 of file adjlistgraph.h.

6.1.3.6 template<unsigned N> const std::vector<EdgeTupleType> IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::vertex_edge_tuples (VIDType *id*) const [inline]

vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Parameters

id	`id`

Returns

:

- `id[0,N)`

Definition at line 130 of file adjlistgraph.h.

6.1.3.7 template<unsigned N> EWeightType IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::weight (VIDType *id_from,* VIDType *id_to*) const `[inline]`

weight:

Parameters

id_from	`id`
id_to	`id`

Returns

:

`id_fromid_tostd::invalid_argument`

- `id_fromid_to >id_fromid_to[0,N)`
- `id_fromid_to`

Definition at line 180 of file adjlistgraph.h.

6.1.4 Member Data Documentation

6.1.4.1 template<unsigned N> std::array<std::vector<std::pair<VIDType,EWeightType> >,N> IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::array

Definition at line 195 of file adjlistgraph.h.

6.1.4.2 template<unsigned N> const unsigned IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >::NUM =N `[static]`

Definition at line 40 of file adjlistgraph.h.

The documentation for this struct was generated from the following file:

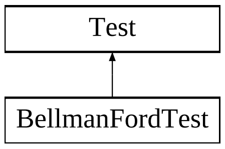
- src/graph_algorithms/basic_graph/graph_representation/adjlist_graph/adjlistgraph.h

6.2 BellmanFordTest Class Reference

BellmanFordTest:

`#include <bellmanford_test.h>`

Inheritance diagram for BellmanFordTest:



Public Types

- typedef Graph< B_NUM, VertexP< int > > GraphType
- typedef VertexP< int > VertexType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GraphType > _1v_graph
- std::shared_ptr< GraphType > _1e_graph
- std::shared_ptr< GraphType > _normal_graph
- std::shared_ptr< GraphType > _minus_graph

6.2.1 Detailed Description

BellmanFordTest:

`BellmanFordTest ::testing::Test TEST_F`

Definition at line 40 of file bellmanford_test.h.

6.2.2 Member Typedef Documentation

6.2.2.1 typedef Graph<B_NUM,VertexP<int> > BellmanFordTest::GraphType

`VertexP<int>`

Definition at line 43 of file bellmanford_test.h.

6.2.2.2 typedef VertexP<int> BellmanFordTest::VertexType

`VertexP<int>`

Definition at line 44 of file bellmanford_test.h.

6.2.3 Member Function Documentation

6.2.3.1 void BellmanFordTest::SetUp ( ) `[inline], [protected]`

Definition at line 46 of file bellmanford_test.h.

6.2.3.2 void BellmanFordTest::TearDown ( ) [inline], [protected]

Definition at line 72 of file bellmanford_test.h.

6.2.4 Member Data Documentation

6.2.4.1 std::shared_ptr< GraphType > BellmanFordTest::_1e_graph [protected]

Definition at line 75 of file bellmanford_test.h.

6.2.4.2 std::shared_ptr< GraphType > BellmanFordTest::_1v_graph [protected]

Definition at line 74 of file bellmanford_test.h.

6.2.4.3 std::shared_ptr< GraphType > BellmanFordTest::_minus_graph [protected]

Definition at line 77 of file bellmanford_test.h.

6.2.4.4 std::shared_ptr< GraphType > BellmanFordTest::_normal_graph [protected]

Definition at line 76 of file bellmanford_test.h.

The documentation for this class was generated from the following file:

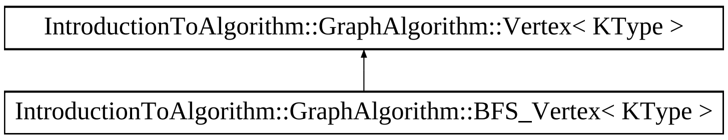
- src/graph_algorithms/single_source_shortest_path/bellman_ford/bellmanford_test.h

6.3 IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType > Struct Template Reference

BFS_Vertex2222.2

`#include <bfs_vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >

Public Types

- enum COLOR { COLOR::WHITE, COLOR::GRAY, COLOR::BLACK }
- typedef int VIDType
- typedef KType KeyType

Public Member Functions

- BFS_Vertex ()

  color
- BFS_Vertex (const KeyType &k)

  key
- BFS_Vertex (const KeyType &k, VIDType d)

  key
- void set_source ()

  set_source
- void set_found (std::shared_ptr< BFS_Vertex > v_parent)

  set_found
- std::string to_string ()

  to_string

Public Attributes

- COLOR color
- std::shared_ptr< BFS_Vertex > parent

6.3.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >

BFS_Vertex2222.2

`VertexVertex`

- `colorBFS_Vertex::COLOR::BLACKBFS_Vertex::COLOR::WHITEBFS_Vertex::COLOR::GRAY`
- `parent:`

- `set_source()`
- `set_found(v_parent)`

Definition at line 42 of file bfs_vertex.h.

6.3.2 Member Typedef Documentation

6.3.2.1 template<typename KType > typedef KType IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >::KeyType

Definition at line 46 of file bfs_vertex.h.

6.3.2.2 template<typename KType > typedef int IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >::VIDType

Definition at line 45 of file bfs_vertex.h.

6.3.3 Member Enumeration Documentation

6.3.3.1 template<typename KType > enum **IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex::COLOR** `[strong]`

Enumerator

WHITE

GRAY

BLACK

Definition at line 47 of file bfs_vertex.h.

6.3.4 Constructor & Destructor Documentation

6.3.4.1 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex**< KType >::**BFS_Vertex** ( ) `[inline]`

`color`

Definition at line 51 of file bfs_vertex.h.

6.3.4.2 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex**< KType >::**BFS_Vertex** (const **KeyType** & *k*) `[inline]`, `[explicit]`

`key`

Parameters

k:	

Definition at line 58 of file bfs_vertex.h.

6.3.4.3 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex**< KType >::**BFS_Vertex** (const **KeyType** & *k*, **VIDType** *d*) `[inline]`

`key`

Parameters

k:	
d:	

Definition at line 66 of file bfs_vertex.h.

6.3.5 Member Function Documentation

6.3.5.1 template<typename KType > void **IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex**< KType >::**set_found** (std::shared_ptr< **BFS_Vertex**< KType > > *v_parent*) `[inline]`

set_found

Parameters

v_parent:	

-
- `parentv_parent`

`v_parentv_parent`

Definition at line 94 of file bfs_vertex.h.

6.3.5.2 template<typename KType > void IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >::set_source ( ) [inline]

set_source

-
- `parent`

Definition at line 76 of file bfs_vertex.h.

6.3.5.3 template<typename KType > std::string IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >::to_string ( ) [inline]

to_string

Returns

:

`Vertexcolorparent`

Definition at line 110 of file bfs_vertex.h.

6.3.6 Member Data Documentation

6.3.6.1 template<typename KType > COLOR IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >::color

Definition at line 133 of file bfs_vertex.h.

6.3.6.2 template<typename KType > std::shared_ptr<BFS_Vertex> IntroductionToAlgorithm::GraphAlgorithm↩ ::BFS_Vertex< KType >::parent

Definition at line 134 of file bfs_vertex.h.

The documentation for this struct was generated from the following file:

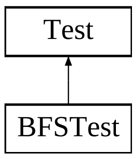
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/bfs_vertex.h

6.4 BFSTest Class Reference

BFSTest:

```
#include <bfs_test.h>
```

Inheritance diagram for BFSTest:



Public Types

- typedef Graph< BFS_N, BFS_Vertex< int > > GType
- typedef std::function< void(BFS_Vertex< int >::VIDType v_id)> ActionType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _list_graph

6.4.1 Detailed Description

BFSTest:

`BFSTest ::testing::Test TEST_F`

Definition at line 41 of file bfs_test.h.

6.4.2 Member Typedef Documentation

6.4.2.1 typedef std::function<void(BFS_Vertex<int>::VIDType v_id)> BFSTest::ActionType

Action

Definition at line 45 of file bfs_test.h.

6.4.2.2 typedef Graph<BFS_N,BFS_Vertex<int> > BFSTest::GType

`BFS_Vertex<int>`

Definition at line 44 of file bfs_test.h.

6.4.3 Member Function Documentation

6.4.3.1 void BFSTest::SetUp ( ) [inline], [protected]

Definition at line 47 of file bfs_test.h.

6.4.3.2 void BFSTest::TearDown ( ) [inline], [protected]

Definition at line 64 of file bfs_test.h.

6.4.4 Member Data Documentation

6.4.4.1 std::shared_ptr<GType> BFSTest::_1e_graph [protected]

Definition at line 66 of file bfs_test.h.

6.4.4.2 std::shared_ptr<GType> BFSTest::_1v_graph [protected]

Definition at line 65 of file bfs_test.h.

6.4.4.3 std::shared_ptr<GType> BFSTest::_list_graph [protected]

Definition at line 67 of file bfs_test.h.

The documentation for this class was generated from the following file:

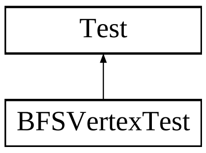
- src/graph_algorithms/basic_graph/graph_bfs/bfs_test.h

6.5 BFSVertexTest Class Reference

BFSVertexTest:

`#include <bfs_vertex_test.h>`

Inheritance diagram for BFSVertexTest:



Public Types

- typedef BFS_Vertex< double > BFS_Vertex

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< BFS_Vertex > _default_vertex
- std::shared_ptr< BFS_Vertex > _normal_vertex

6.5.1 Detailed Description

BFSVertexTest:

`BFSVertexTest ::testing::Test TEST_F`

Definition at line 31 of file bfs_vertex_test.h.

6.5.2 Member Typedef Documentation

6.5.2.1 typedef **BFS_Vertex**<double> **BFSVertexTest::BFS_Vertex**

`BFS_Vertex<double>`

Definition at line 34 of file bfs_vertex_test.h.

6.5.3 Member Function Documentation

6.5.3.1 void BFSVertexTest::SetUp ( ) `[inline], [protected]`

Definition at line 37 of file bfs_vertex_test.h.

6.5.3.2 void BFSVertexTest::TearDown ( ) `[inline], [protected]`

Definition at line 41 of file bfs_vertex_test.h.

6.5.4 Member Data Documentation

6.5.4.1 std::shared_ptr<**BFS_Vertex**> BFSVertexTest::_default_vertex `[protected]`

Definition at line 42 of file bfs_vertex_test.h.

6.5.4.2 std::shared_ptr<**BFS_Vertex**> BFSVertexTest::_normal_vertex `[protected]`

Definition at line 43 of file bfs_vertex_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/bfs_vertex_test.h

6.6 IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT > Struct Template Reference

BinaryTree1010.4

`#include <binarytree.h>`

Public Types

- typedef NodeT NodeType
- typedef NodeT::KeyType KeyType

Public Member Functions

- BinaryTree ()
- std::string to_xml ()

  to_xml:`xml`

Public Attributes

- std::shared_ptr< NodeType > root

6.6.1 Detailed Description

template<typename NodeT>struct IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >

BinaryTree1010.4

rootroot

Definition at line 32 of file binarytree.h.

6.6.2 Member Typedef Documentation

6.6.2.1 template<typename NodeT> typedef NodeT::KeyType IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >::KeyType

Definition at line 35 of file binarytree.h.

6.6.2.2 template<typename NodeT> typedef NodeT IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >::NodeType

Definition at line 34 of file binarytree.h.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 template<typename NodeT> IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >::BinaryTree ( ) `[inline]`

Definition at line 41 of file binarytree.h.

6.6.4 Member Function Documentation

6.6.4.1 template<typename NodeT> std::string IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >::to_xml ( ) [inline]

to_xml:`xml`

Returns

: `xml`

`xml`

Definition at line 51 of file binarytree.h.

6.6.5 Member Data Documentation

6.6.5.1 template<typename NodeT> std::shared_ptr<NodeType> IntroductionToAlgorithm::TreeAlgorithm::↩ BinaryTree< NodeT >::root

Definition at line 59 of file binarytree.h.

The documentation for this struct was generated from the following file:

- src/tree_algorithms/binarytree/binarytree.h

6.7 IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType > Struct Template Reference

BinaryTreeNodexxxx

`#include <binarytreenode.h>`

Public Types

- typedef KType KeyType

Public Member Functions

- BinaryTreeNode ()
- BinaryTreeNode (const KeyType &keyvalue)
- virtual std::string to_string ()

  to_string:
- virtual std::string to_xml ()

  to_xml:`xml`
- bool is_left_child ()

  is_left_child:
- bool is_right_child ()

  is_right_child:

Public Attributes

- std::weak_ptr< BinaryTreeNode > parent
- std::shared_ptr< BinaryTreeNode > lchild
- std::shared_ptr< BinaryTreeNode > rchild
- KeyType key

6.7.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >

BinaryTreeNodexxxx

`key`

Definition at line 31 of file binarytreenode.h.

6.7.2 Member Typedef Documentation

6.7.2.1 template<typename KType > typedef KType IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::KeyType

Definition at line 34 of file binarytreenode.h.

6.7.3 Constructor & Destructor Documentation

6.7.3.1 template<typename KType > IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::BinaryTreeNode ( ) `[inline]`

Definition at line 39 of file binarytreenode.h.

6.7.3.2 template<typename KType > IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::BinaryTreeNode (const KeyType & *keyvalue*) `[inline]`, `[explicit]`

Parameters

keyvalue:	`key`

Definition at line 48 of file binarytreenode.h.

6.7.4 Member Function Documentation

6.7.4.1 template<typename KType > bool IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::is_left_child ( ) `[inline]`

is_left_child:

Returns

`truefalse falsetruefalse`

Definition at line 99 of file binarytreenode.h.

6.7.4.2 template<typename KType > bool IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::is_right_child ( ) [inline]

is_right_child:

Returns

`truefalse` `falsetruefalse`

Definition at line 113 of file binarytreenode.h.

6.7.4.3 template<typename KType > virtual std::string IntroductionToAlgorithm::TreeAlgorithm::BinaryTree←Node< KType >::to_string ( ) [inline], [virtual]

to_string:

Returns

:

`keykey`

Definition at line 58 of file binarytreenode.h.

6.7.4.4 template<typename KType > virtual std::string IntroductionToAlgorithm::TreeAlgorithm::BinaryTree←Node< KType >::to_xml ( ) [inline], [virtual]

to_xml:`xml`

Returns

: `xml`

`xmlxml`

Definition at line 79 of file binarytreenode.h.

6.7.5 Member Data Documentation

6.7.5.1 template<typename KType > KeyType IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >::key

Definition at line 125 of file binarytreenode.h.

6.7.5.2 template<typename KType > std::shared_ptr<BinaryTreeNode> IntroductionToAlgorithm::Tree←Algorithm::BinaryTreeNode< KType >::lchild

Definition at line 123 of file binarytreenode.h.

6.7.5.3 template<typename KType > std::weak_ptr<BinaryTreeNode> IntroductionToAlgorithm::TreeAlgorithm←::BinaryTreeNode< KType >::parent

Definition at line 122 of file binarytreenode.h.

6.7.5.4 template< typename KType > std::shared_ptr< BinaryTreeNode > IntroductionToAlgorithm::Tree↩ Algorithm::BinaryTreeNode< KType >::rchild

Definition at line 124 of file binarytreenode.h.

The documentation for this struct was generated from the following file:

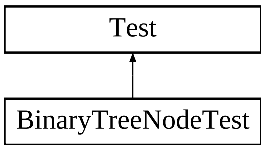
- src/tree_algorithms/binarytreenode/binarytreenode.h

6.8 BinaryTreeNodeTest Class Reference

BinaryTreeNodeTest:

`#include <binarytreenode_test.h>`

Inheritance diagram for BinaryTreeNodeTest:



Public Types

- typedef BinaryTreeNode< int > Node

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< Node > default_node
- std::shared_ptr< Node > root_node

6.8.1 Detailed Description

BinaryTreeNodeTest:

`BinaryTreeNodeTest ::testing::Test TEST_F`

Definition at line 29 of file binarytreenode_test.h.

6.8.2 Member Typedef Documentation

6.8.2.1 typedef BinaryTreeNode<int> BinaryTreeNodeTest::Node

Definition at line 32 of file binarytreenode_test.h.

6.8.3 Member Function Documentation

6.8.3.1 void BinaryTreeNodeTest::SetUp ( ) [inline], [protected]

Definition at line 34 of file binarytreenode_test.h.

6.8.3.2 void BinaryTreeNodeTest::TearDown ( ) [inline], [protected]

Definition at line 43 of file binarytreenode_test.h.

6.8.4 Member Data Documentation

6.8.4.1 std::shared_ptr<Node> BinaryTreeNodeTest::default_node [protected]

Definition at line 46 of file binarytreenode_test.h.

6.8.4.2 std::shared_ptr<Node> BinaryTreeNodeTest::root_node [protected]

Definition at line 47 of file binarytreenode_test.h.

The documentation for this class was generated from the following file:

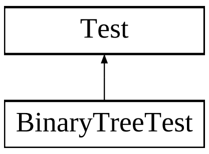
- src/tree_algorithms/binarytreenode/binarytreenode_test.h

6.9 BinaryTreeTest Class Reference

BinaryTreeTest:

`#include <binarytree_test.h>`

Inheritance diagram for BinaryTreeTest:



Public Types

- typedef BinaryTreeNode< int > Node

Protected Member Functions

- BinaryTreeTest ()
- void SetUp ()

  SetUp:
- void TearDown ()

  TearDown:

Protected Attributes

- BinaryTree< Node > _empty_tree
- BinaryTree< Node > _normal_tree

6.9.1 Detailed Description

BinaryTreeTest:

`BinaryTreeTest ::testing::Test TEST_F`

Definition at line 37 of file binarytree_test.h.

6.9.2 Member Typedef Documentation

6.9.2.1 typedef BinaryTreeNode<int> BinaryTreeTest::Node

Definition at line 40 of file binarytree_test.h.

6.9.3 Constructor & Destructor Documentation

6.9.3.1 BinaryTreeTest::BinaryTreeTest ( ) `[inline]`, `[protected]`

Definition at line 42 of file binarytree_test.h.

6.9.4 Member Function Documentation

6.9.4.1 void BinaryTreeTest::SetUp ( ) `[inline]`, `[protected]`

SetUp:

`SetUp ::testing::Test`

Definition at line 49 of file binarytree_test.h.

6.9.4.2 void BinaryTreeTest::TearDown ( ) `[inline]`, `[protected]`

TearDown:

`TearDown ::testing::Test`

Definition at line 89 of file binarytree_test.h.

6.9.5 Member Data Documentation

6.9.5.1 BinaryTree<Node> BinaryTreeTest::_empty_tree `[protected]`

Definition at line 91 of file binarytree_test.h.

6.9.5.2 BinaryTree<Node> BinaryTreeTest::_normal_tree `[protected]`

Definition at line 92 of file binarytree_test.h.

The documentation for this class was generated from the following file:

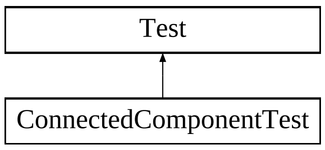
- src/tree_algorithms/binarytree/binarytree_test.h

6.10 ConnectedComponentTest Class Reference

ConnectedComponentTest:

`#include <connectedcomponent_test.h>`

Inheritance diagram for ConnectedComponentTest:



Public Types

- typedef Graph< C_NUM, SetVertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.10.1 Detailed Description

ConnectedComponentTest:

`ConnectedComponentTest ::testing::Test TEST_F`

Definition at line 40 of file connectedcomponent_test.h.

6.10.2 Member Typedef Documentation

6.10.2.1 typedef Graph<C_NUM,SetVertex<int> > ConnectedComponentTest::GType

`SetVertex<int>`

Definition at line 43 of file connectedcomponent_test.h.

6.10.3 Member Function Documentation

6.10.3.1 void ConnectedComponentTest::SetUp ( ) [inline], [protected]

Definition at line 45 of file connectedcomponent_test.h.

6.10.3.2 void ConnectedComponentTest::TearDown ( ) [inline], [protected]

Definition at line 58 of file connectedcomponent_test.h.

6.10.4 Member Data Documentation

6.10.4.1 std::shared_ptr< GType > ConnectedComponentTest::_graph [protected]

Definition at line 60 of file connectedcomponent_test.h.

The documentation for this class was generated from the following file:

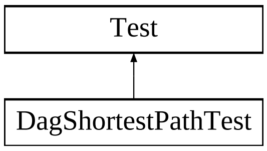
- src/graph_algorithms/basic_graph/connected_component/connectedcomponent_test.h

6.11 DagShortestPathTest Class Reference

DagShortestPathTest:

`#include <dagshortpath_test.h>`

Inheritance diagram for DagShortestPathTest:



Public Types

- typedef Graph< DSP_NUM, DFS_Vertex< int > > GType
- typedef DFS_Vertex< int > VertexType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _normal_graph

6.11.1 Detailed Description

DagShortestPathTest:

`DagShortestPathTest ::testing::Test TEST_F`

Definition at line 39 of file dagshortpath_test.h.

6.11.2 Member Typedef Documentation

6.11.2.1 typedef **Graph**< DSP_NUM,**DFS_Vertex**< int> > **DagShortestPathTest::GType**

`DFS_Vertex<int>`

Definition at line 42 of file dagshortpath_test.h.

6.11.2.2 typedef **DFS_Vertex**< int> **DagShortestPathTest::VertexType**

`DFS_Vertex<int>`

Definition at line 43 of file dagshortpath_test.h.

6.11.3 Member Function Documentation

6.11.3.1 void DagShortestPathTest::SetUp ( ) `[inline]`, `[protected]`

Definition at line 45 of file dagshortpath_test.h.

6.11.3.2 void DagShortestPathTest::TearDown ( ) `[inline]`, `[protected]`

Definition at line 63 of file dagshortpath_test.h.

6.11.4 Member Data Documentation

6.11.4.1 std::shared_ptr< **GType**> DagShortestPathTest::_1e_graph `[protected]`

Definition at line 66 of file dagshortpath_test.h.

6.11.4.2 std::shared_ptr< **GType**> DagShortestPathTest::_1v_graph `[protected]`

Definition at line 65 of file dagshortpath_test.h.

6.11.4.3 std::shared_ptr< **GType**> DagShortestPathTest::_normal_graph `[protected]`

Definition at line 67 of file dagshortpath_test.h.

The documentation for this class was generated from the following file:

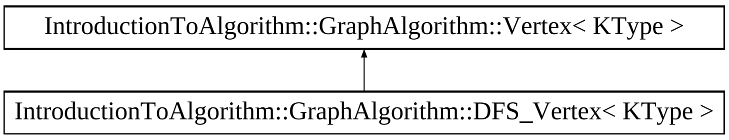
- src/graph_algorithms/single_source_shortest_path/dag_shortest_path/dagshortpath_test.h

6.12 IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType > Struct Template Reference

DFS_Vertex2222.3

`#include <dfs_vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >

Public Types

- enum COLOR { COLOR::WHITE, COLOR::GRAY, COLOR::BLACK }
- typedef int VIDType
- typedef KType KeyType

Public Member Functions

- DFS_Vertex ()

  `color`-1
- DFS_Vertex (const KeyType &k)

  `key`
- DFS_Vertex (const KeyType &k, VIDType d)

  `key`
- void set_disovered (int discover_t)

  set_disovered
- void set_finished (int finish_t)

  set_finished
- std::string to_string ()

  to_string

Public Attributes

- int discover_time
- int finish_time
- COLOR color
- std::shared_ptr< DFS_Vertex > parent

6.12.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >

DFS_Vertex2222.3

`VertexVertex`

- `colorDFS_Vertex::COLOR::BLACKDFS_Vertex::COLOR::WHITEDFS_Vertex::COLOR↩ ::GRAY`
- `parent:`
- `discover_time`
- `finish_time`

- `set_disovered(discover_t)`
- `set_finished(finish_t)`

Definition at line 42 of file dfs_vertex.h.

6.12.2 Member Typedef Documentation

6.12.2.1 template<typename KType> typedef KType **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::KeyType**

Definition at line 46 of file dfs_vertex.h.

6.12.2.2 template<typename KType> typedef int **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::VIDType**

Definition at line 45 of file dfs_vertex.h.

6.12.3 Member Enumeration Documentation

6.12.3.1 template<typename KType> enum **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex::COLOR** `[strong]`

Enumerator

WHITE
GRAY
BLACK

Definition at line 47 of file dfs_vertex.h.

6.12.4 Constructor & Destructor Documentation

6.12.4.1 template<typename KType> **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::DFS_Vertex ( )** `[inline]`

`color`-1

Definition at line 51 of file dfs_vertex.h.

6.12.4.2 template<typename KType> **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::DFS_Vertex (const KeyType & *k*)** `[inline]`, `[explicit]`

`key`

Parameters

k:	-1

Definition at line 58 of file dfs_vertex.h.

6.12.4.3 template<typename KType> **IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::DFS_Vertex (const KeyType & *k*, VIDType *d*)** `[inline]`

`key`

Parameters

k:	
d:	-1

Definition at line 68 of file dfs_vertex.h.

6.12.5 Member Function Documentation

6.12.5.1 template<typename KType> void IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::set_disovered (int *discover_t*) [inline]

set_disovered

Parameters

discover_t:	

-
- `discover_timediscover_t`

Definition at line 81 of file dfs_vertex.h.

6.12.5.2 template<typename KType> void IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::set_finished (int *finish_t*) [inline]

set_finished

Parameters

finish_t:	

-
- `finish_timefinish_t`

Definition at line 96 of file dfs_vertex.h.

6.12.5.3 template<typename KType> std::string IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::to_string ( ) [inline]

to_string

Returns

:

`Vertexcolorparentdiscover_timefinish_time`

Definition at line 107 of file dfs_vertex.h.

6.12.6 Member Data Documentation

6.12.6.1 template<typename KType> COLOR IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::color

Definition at line 132 of file dfs_vertex.h.

6.12.6.2 template< typename KType > int IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::discover_time

Definition at line 130 of file dfs_vertex.h.

6.12.6.3 template< typename KType > int IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >::finish_time

Definition at line 131 of file dfs_vertex.h.

6.12.6.4 template< typename KType > std::shared_ptr< DFS_Vertex > IntroductionToAlgorithm::GraphAlgorithm↩ ::DFS_Vertex< KType >::parent

Definition at line 133 of file dfs_vertex.h.

The documentation for this struct was generated from the following file:

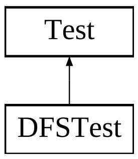
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/dfs_vertex.h

6.13 DFSTest Class Reference

DFSTest:

`#include <dfs_test.h>`

Inheritance diagram for DFSTest:



Public Types

- typedef Graph< DFS_N, DFS_Vertex< double > > GType
- typedef std::function< void(DFS_Vertex< double >::VIDType v_id, int time)> ActionType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _list_graph
- std::shared_ptr< GType > _rlist_graph

6.13.1 Detailed Description

DFSTest:

`DFSTest ::testing::Test TEST_F`

Definition at line 42 of file dfs_test.h.

6.13.2 Member Typedef Documentation

6.13.2.1 typedef std::function<void(DFS_Vertex<double>::VIDType v_id,int time)> DFSTest::ActionType

Action

Definition at line 46 of file dfs_test.h.

6.13.2.2 typedef Graph<DFS_N,DFS_Vertex<double> > DFSTest::GType

`DFS_Vertex<double>`

Definition at line 45 of file dfs_test.h.

6.13.3 Member Function Documentation

6.13.3.1 void DFSTest::SetUp ( ) `[inline]`, `[protected]`

Definition at line 48 of file dfs_test.h.

6.13.3.2 void DFSTest::TearDown ( ) `[inline]`, `[protected]`

Definition at line 73 of file dfs_test.h.

6.13.4 Member Data Documentation

6.13.4.1 std::shared_ptr<GType> DFSTest::_1e_graph `[protected]`

Definition at line 75 of file dfs_test.h.

6.13.4.2 std::shared_ptr<GType> DFSTest::_1v_graph `[protected]`

Definition at line 74 of file dfs_test.h.

6.13.4.3 std::shared_ptr<GType> DFSTest::_list_graph `[protected]`

Definition at line 76 of file dfs_test.h.

6.13.4.4 std::shared_ptr<GType> DFSTest::_rlist_graph `[protected]`

Definition at line 77 of file dfs_test.h.

The documentation for this class was generated from the following file:

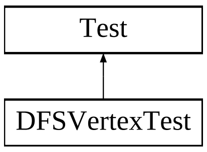
- src/graph_algorithms/basic_graph/graph_dfs/dfs_test.h

6.14 DFSVertexTest Class Reference

DFSVertexTest:

```
#include <dfs_vertex_test.h>
```

Inheritance diagram for DFSVertexTest:



Public Types

- typedef DFS_Vertex< double > DFS_Vertex

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< DFS_Vertex > _default_vertex
- std::shared_ptr< DFS_Vertex > _normal_vertex

6.14.1 Detailed Description

DFSVertexTest:

```
DFSVertexTest ::testing::Test TEST_F
```

Definition at line 30 of file dfs_vertex_test.h.

6.14.2 Member Typedef Documentation

6.14.2.1 typedef DFS_Vertex<double> DFSVertexTest::DFS_Vertex

```
DFS_Vertex<double>
```

Definition at line 33 of file dfs_vertex_test.h.

6.14.3 Member Function Documentation

6.14.3.1 void DFSVertexTest::SetUp ( ) [inline], [protected]

Definition at line 36 of file dfs_vertex_test.h.

6.14.3.2 void DFSVertexTest::TearDown ( ) [inline], [protected]

Definition at line 41 of file dfs_vertex_test.h.

6.14.4 Member Data Documentation

6.14.4.1 std::shared_ptr<DFS_Vertex> DFSVertexTest::_default_vertex [protected]

Definition at line 42 of file dfs_vertex_test.h.

6.14.4.2 std::shared_ptr<DFS_Vertex> DFSVertexTest::_normal_vertex [protected]

Definition at line 43 of file dfs_vertex_test.h.

The documentation for this class was generated from the following file:

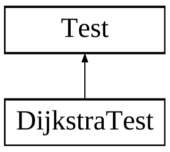
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/dfs_vertex_test.h

6.15 DijkstraTest Class Reference

DijkstraTest:

`#include <dijkstra_test.h>`

Inheritance diagram for DijkstraTest:



Public Types

- typedef Graph< DIJK_NUM, VertexP< int > > GraphType
- typedef VertexP< int > VertexType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GraphType > _1v_graph
- std::shared_ptr< GraphType > _1e_graph
- std::shared_ptr< GraphType > _normal_graph

6.15.1 Detailed Description

DijkstraTest:

`DijkstraTest ::testing::Test TEST_F`

Definition at line 40 of file dijkstra_test.h.

6.15.2 Member Typedef Documentation

6.15.2.1 typedef Graph<DIJK_NUM,VertexP<int> > DijkstraTest::GraphType

`VertexP<int>`

Definition at line 43 of file dijkstra_test.h.

6.15.2.2 typedef VertexP<int> DijkstraTest::VertexType

`VertexP<int>`

Definition at line 44 of file dijkstra_test.h.

6.15.3 Member Function Documentation

6.15.3.1 void DijkstraTest::SetUp ( ) `[inline], [protected]`

Definition at line 46 of file dijkstra_test.h.

6.15.3.2 void DijkstraTest::TearDown ( ) `[inline], [protected]`

Definition at line 64 of file dijkstra_test.h.

6.15.4 Member Data Documentation

6.15.4.1 std::shared_ptr<GraphType> DijkstraTest::_1e_graph `[protected]`

Definition at line 67 of file dijkstra_test.h.

6.15.4.2 std::shared_ptr<GraphType> DijkstraTest::_1v_graph `[protected]`

Definition at line 66 of file dijkstra_test.h.

6.15.4.3 std::shared_ptr<GraphType> DijkstraTest::_normal_graph `[protected]`

Definition at line 68 of file dijkstra_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/single_source_shortest_path/dijkstra/dijkstra_test.h

6.16 IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType > Struct Template Reference

DisjointSetNode2121.3

`#include <disjointset.h>`

Public Types

- typedef KType KeyType

Public Member Functions

- DisjointSetNode (std::shared_ptr< KeyType >v)

    DisjointSetNode

Static Public Member Functions

- static std::shared_ptr< DisjointSetNode > find_set (std::shared_ptr< DisjointSetNode >node)

    find_set
- static void make_set (std::shared_ptr< DisjointSetNode >node)

    make_set
- static void link_set (std::shared_ptr< DisjointSetNode >nodeX, std::shared_ptr< DisjointSetNode >nodeY)

    link_set
- static void union_set (std::shared_ptr< DisjointSetNode >nodeX, std::shared_ptr< DisjointSetNode >nodeY)

    union_set

Public Attributes

- std::weak_ptr< KeyType > value
- int rank
- std::shared_ptr< DisjointSetNode > parent

6.16.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >

DisjointSetNode2121.3

{S1,S2,...Sk} x,y)

- make_set(x):x
- unionx(x,y):xy(SxSy)SxSy
- find_set(x):x

- xrank,xxunion
- `find_set`

O(m∗alpha∗n))alpha(n) alpha(n)<=4nm

Definition at line 54 of file disjointset.h.

6.16.2 Member Typedef Documentation

6.16.2.1 template<typename KType > typedef KType IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::KeyType

Definition at line 56 of file disjointset.h.

6.16.3 Constructor & Destructor Documentation

6.16.3.1 template<typename KType > IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::DisjointSetNode (std::shared_ptr< KeyType > *v*) [inline], [explicit]

DisjointSetNode

Parameters

v:	vvalue

Definition at line 65 of file disjointset.h.

6.16.4 Member Function Documentation

6.16.4.1 template<typename KType > static std::shared_ptr<DisjointSetNode> IntroductionToAlgorithm::Set←- Algorithm::DisjointSetNode< KType >::find_set (std::shared_ptr< DisjointSetNode< KType > > *node*) [inline], [static]

find_set

Parameters

node:	

Returns

:

find_set two_pass method

Definition at line 84 of file disjointset.h.

6.16.4.2 template<typename KType > static void IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::link_set (std::shared_ptr< DisjointSetNode< KType > > *nodeX,* std::shared_ptr< DisjointSetNode< KType > > *nodeY*) [inline], [static]

link_set

Parameters

nodeX:	
nodeY:	xrank,xx.

- nodeXnodeY
- nodeX nodeYnodeXnodeY

Definition at line 124 of file disjointset.h.

6.16.4.3 template< typename KType > static void IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::make_set (std::shared_ptr< DisjointSetNode< KType > > *node*) [inline], [static]

make_set

Parameters

node:	0

Definition at line 100 of file disjointset.h.

6.16.4.4 template< typename KType > static void IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::union_set (std::shared_ptr< DisjointSetNode< KType > > *nodeX*, std::shared_ptr< DisjointSetNode< KType > > *nodeY*) [inline], [static]

union_set

Parameters

nodeX:	
nodeY:	

Definition at line 151 of file disjointset.h.

6.16.5 Member Data Documentation

6.16.5.1 template< typename KType > std::shared_ptr<DisjointSetNode> IntroductionToAlgorithm::Set↩ Algorithm::DisjointSetNode< KType >::parent

Definition at line 69 of file disjointset.h.

6.16.5.2 template< typename KType > int IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >::rank

Definition at line 68 of file disjointset.h.

6.16.5.3 template< typename KType > std::weak_ptr<KeyType> IntroductionToAlgorithm::SetAlgorithm::↩ DisjointSetNode< KType >::value

Definition at line 67 of file disjointset.h.

The documentation for this struct was generated from the following file:

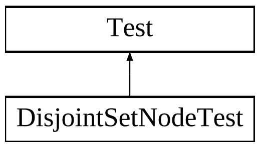
- src/set_algorithms/disjoint_set/disjointset.h

6.17 DisjointSetNodeTest Class Reference

DisjointSetNodeTest:

`#include <disjointset_test.h>`

Inheritance diagram for DisjointSetNodeTest:



Public Types

- typedef DisjointSetNode< int > NodeType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< NodeType > nodes [S_NUM]

6.17.1 Detailed Description

DisjointSetNodeTest:

`DisjointSetNodeTest ::testing::Test TEST_F`

Definition at line 33 of file disjointset_test.h.

6.17.2 Member Typedef Documentation

6.17.2.1 typedef DisjointSetNode<int> DisjointSetNodeTest::NodeType

Definition at line 36 of file disjointset_test.h.

6.17.3 Member Function Documentation

6.17.3.1 void DisjointSetNodeTest::SetUp ( ) `[inline], [protected]`

Definition at line 38 of file disjointset_test.h.

6.17.3.2 void DisjointSetNodeTest::TearDown ( ) `[inline], [protected]`

Definition at line 43 of file disjointset_test.h.

6.17.4 Member Data Documentation

6.17.4.1 std::shared_ptr<NodeType> DisjointSetNodeTest::nodes[S_NUM] `[protected]`

Definition at line 44 of file disjointset_test.h.

The documentation for this class was generated from the following file:

- src/set_algorithms/disjoint_set/disjointset_test.h

6.18 IntroductionToAlgorithm::GraphAlgorithm::Edge< VType > Struct Template Reference

Edge2222.1

```
#include <edge.h>
```

Public Types

- typedef int VIDType
- typedef int EWeightType
- typedef VType VertexType
- typedef std::tuple< VIDType, VIDType, EWeightType > EdgeTupleType

Public Member Functions

- Edge (std::shared_ptr< VertexType >v1, std::shared_ptr< VertexType >v2, EWeightType w)

- virtual std::string to_string () const

  to_string
- const EdgeTupleType edge_tuple () const

  edge_tuple

Public Attributes

- std::shared_ptr< VertexType > vertex1
- std::shared_ptr< VertexType > vertex2
- EWeightType weight

6.18.1 Detailed Description

template< typename VType >struct IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >

Edge2222.1

- `vertex1`
- `vertex2`
- `weight`: int

`std::tuple<VIDType,VIDType,EWeightType>idid`

Definition at line 39 of file edge.h.

6.18.2 Member Typedef Documentation

6.18.2.1 template< typename VType > typedef std::tuple< VIDType,VIDType,EWeightType> IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::EdgeTupleType

12)

Definition at line 45 of file edge.h.

6.18.2.2 template< typename VType > typedef int IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::EWeightType

Definition at line 43 of file edge.h.

6.18.2.3 template<typename VType > typedef VType IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::VertexType

Definition at line 44 of file edge.h.

6.18.2.4 template<typename VType > typedef int IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::VIDType

Definition at line 42 of file edge.h.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 template<typename VType > IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::Edge (std::shared_ptr< VertexType > *v1*, std::shared_ptr< VertexType > *v2*, EWeightType *w*) `[inline]`

Parameters

v1:	
v2:	
w	

Definition at line 52 of file edge.h.

6.18.4 Member Function Documentation

6.18.4.1 template<typename VType > const EdgeTupleType IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::edge_tuple ( ) const `[inline]`

edge_tuple

Returns

:

Definition at line 72 of file edge.h.

6.18.4.2 template<typename VType > virtual std::string IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::to_string ( ) const `[inline]`, `[virtual]`

to_string

Returns

:

Definition at line 62 of file edge.h.

6.18.5 Member Data Documentation

6.18.5.1 template<typename VType > std::shared_ptr<VertexType> IntroductionToAlgorithm::GraphAlgorithm↩ ::Edge< VType >::vertex1

Definition at line 76 of file edge.h.

6.18.5.2 template<typename VType > std::shared_ptr<VertexType> IntroductionToAlgorithm::GraphAlgorithm↩ ::Edge< VType >::vertex2

Definition at line 77 of file edge.h.

6.18.5.3 template<typename VType > EWeightType IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >::weight

Definition at line 78 of file edge.h.

The documentation for this struct was generated from the following file:

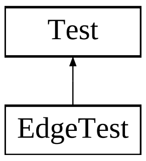
- src/graph_algorithms/basic_graph/graph_representation/graph_edge/edge.h

6.19 EdgeTest Class Reference

EdgeTest:`Edge`

`#include <edge_test.h>`

Inheritance diagram for EdgeTest:



Public Types

- typedef Vertex< double > Node

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< Edge< Node > > _edge

6.19.1 Detailed Description

EdgeTest:`Edge`

`EdgeTest ::testing::Test TEST_F`

Definition at line 32 of file edge_test.h.

6.19.2 Member Typedef Documentation

6.19.2.1 typedef Vertex<double> EdgeTest::Node

Definition at line 35 of file edge_test.h.

6.19.3 Member Function Documentation

6.19.3.1 void EdgeTest::SetUp ( ) [inline], [protected]

Definition at line 37 of file edge_test.h.

6.19.3.2 void EdgeTest::TearDown ( ) [inline], [protected]

Definition at line 40 of file edge_test.h.

6.19.4 Member Data Documentation

6.19.4.1 std::shared_ptr<Edge<Node> > EdgeTest::_edge [protected]

Definition at line 42 of file edge_test.h.

The documentation for this class was generated from the following file:

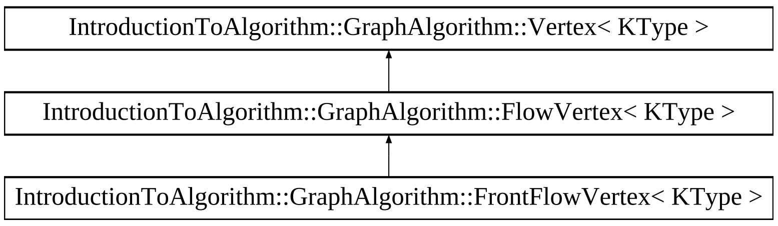
- src/graph_algorithms/basic_graph/graph_representation/graph_edge/edge_test.h

6.20 IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType > Struct Template Reference

FlowVertex-2626.4

`#include <flow_vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >

Public Types

- typedef KType KeyType
- typedef int VIDType

Public Member Functions

- FlowVertex ()

- FlowVertex (const KeyType &k)

  key
- FlowVertex (const KeyType &k, VIDType d)

  key
- virtual std::string to_string () const

  to_string

Public Attributes

- int h

6.20.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >

FlowVertex-2626.4

FlowVertex VertexVertex`int h`

- `KType key`
- `int h`

Definition at line 34 of file flow_vertex.h.

6.20.2 Member Typedef Documentation

6.20.2.1 template<typename KType> typedef KType IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::KeyType

Definition at line 36 of file flow_vertex.h.

6.20.2.2 template<typename KType> typedef int IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::VIDType

Definition at line 37 of file flow_vertex.h.

6.20.3 Constructor & Destructor Documentation

6.20.3.1 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::FlowVertex ( ) `[inline]`

Definition at line 40 of file flow_vertex.h.

6.20.3.2 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::FlowVertex (const KeyType & *k*) `[inline]`, `[explicit]`

`key`

Parameters

k:	

Definition at line 45 of file flow_vertex.h.

6.20.3.3 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::FlowVertex (const KeyType & *k*, VIDType *d*) [inline]

`key`

Parameters

k:	
d:	

Definition at line 51 of file flow_vertex.h.

6.20.4 Member Function Documentation

6.20.4.1 template<typename KType> virtual std::string IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::to_string ( ) const [inline], [virtual]

to_string

Returns

:

`Vertexh`

Reimplemented from IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >.

Reimplemented in IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >.

Definition at line 59 of file flow_vertex.h.

6.20.5 Member Data Documentation

6.20.5.1 template<typename KType> int IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >::h

Definition at line 65 of file flow_vertex.h.

The documentation for this struct was generated from the following file:

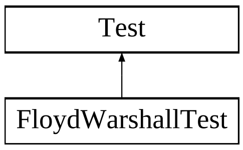
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/flow_vertex.h

6.21 FloydWarshallTest Class Reference

FloydWarshallTest:

`#include <floyd_warshall_test.h>`

Inheritance diagram for FloydWarshallTest:



Public Types

- typedef Graph< FW_N, Vertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.21.1 Detailed Description

FloydWarshallTest:

`FloydWarshallTest ::testing::Test TEST_F`

Definition at line 40 of file floyd_warshall_test.h.

6.21.2 Member Typedef Documentation

6.21.2.1 typedef Graph<FW_N,Vertex<int> > FloydWarshallTest::GType

`Vertex<int>`

Definition at line 43 of file floyd_warshall_test.h.

6.21.3 Member Function Documentation

6.21.3.1 void FloydWarshallTest::SetUp ( ) [inline], [protected]

Definition at line 46 of file floyd_warshall_test.h.

6.21.3.2 void FloydWarshallTest::TearDown ( ) [inline], [protected]

Definition at line 64 of file floyd_warshall_test.h.

6.21.4 Member Data Documentation

6.21.4.1 std::shared_ptr<GType> FloydWarshallTest::_graph [protected]

25-1

Definition at line 65 of file floyd_warshall_test.h.

The documentation for this class was generated from the following file:

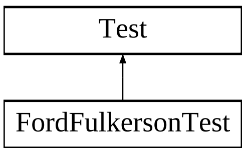
- src/graph_algorithms/all_node_pair_shortest_path/floyd_warshall/floyd_warshall_test.h

6.22 FordFulkersonTest Class Reference

FordFulkersonTest:

`#include <fordfulkerson_test.h>`

Inheritance diagram for FordFulkersonTest:



Public Types

- typedef Graph< FF_N, BFS_Vertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.22.1 Detailed Description

FordFulkersonTest:

`FordFulkersonTest ::testing::Test TEST_F`

Definition at line 39 of file fordfulkerson_test.h.

6.22.2 Member Typedef Documentation

6.22.2.1 typedef **Graph**<**FF_N**,**BFS_Vertex**<int> > **FordFulkersonTest::GType**

`BFS_Vertex<int>`

Definition at line 42 of file fordfulkerson_test.h.

6.22.3 Member Function Documentation

6.22.3.1 void FordFulkersonTest::SetUp ( ) [inline], [protected]

Definition at line 45 of file fordfulkerson_test.h.

6.22.3.2 void FordFulkersonTest::TearDown ( ) [inline], [protected]

Definition at line 63 of file fordfulkerson_test.h.

6.22.4 Member Data Documentation

6.22.4.1 std::shared_ptr< GType > FordFulkersonTest::_graph [protected]

26-6

Definition at line 64 of file fordfulkerson_test.h.

The documentation for this class was generated from the following file:

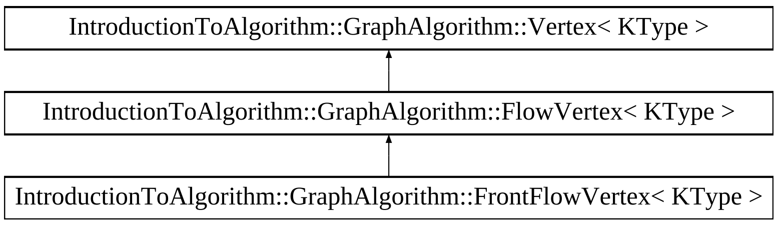
- src/graph_algorithms/max_flow/ford_fulkerson/fordfulkerson_test.h

6.23 IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType > Struct Template Reference

FrontFlowVertexrelabel_to_front2626.4

`#include <front_flow_vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >

Public Types

- typedef KType KeyType
- typedef int VIDType

Public Member Functions

- FrontFlowVertex ()
- FrontFlowVertex (const KeyType &k)

    key
- FrontFlowVertex (const KeyType &k, VIDType d)

    key
- virtual std::string to_string () const

    to_string

Public Attributes

- List< ListNode< FrontFlowVertex > > N_List

6.23.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >

FrontFlowVertexrelabel_to_front2626.4

FrontFlowVertex FlowVertexFlowVertex`N_List`

relabel_to_front FrontFlowVertex

- L L
- u.N u

Definition at line 175 of file front_flow_vertex.h.

6.23.2 Member Typedef Documentation

6.23.2.1 template<typename KType > typedef KType **IntroductionToAlgorithm::GraphAlgorithm::FrontFlow↩ Vertex< KType >::KeyType**

Definition at line 177 of file front_flow_vertex.h.

6.23.2.2 template<typename KType > typedef int **IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >::VIDType**

Definition at line 178 of file front_flow_vertex.h.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >::FrontFlowVertex ( )** `[inline]`

Definition at line 183 of file front_flow_vertex.h.

6.23.3.2 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >::FrontFlowVertex (const KeyType & *k*)** `[inline]`, `[explicit]`

`key`

Parameters

k:	

Definition at line 188 of file front_flow_vertex.h.

6.23.3.3 template<typename KType > **IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >::FrontFlowVertex (const KeyType & *k*, VIDType *d*)** `[inline]`

`key`

Parameters

k:	
d:	

Definition at line 194 of file front_flow_vertex.h.

6.23.4 Member Function Documentation

6.23.4.1 template<typename KType > virtual std::string IntroductionToAlgorithm::GraphAlgorithm::FrontFlow↩Vertex< KType >::to_string ( ) const [inline], [virtual]

to_string

Returns

:

`FlowVertexN_List`

Reimplemented from IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >.

Definition at line 202 of file front_flow_vertex.h.

6.23.5 Member Data Documentation

6.23.5.1 template<typename KType > List<ListNode<FrontFlowVertex> > IntroductionToAlgorithm::Graph↩Algorithm::FrontFlowVertex< KType >::N_List

Definition at line 180 of file front_flow_vertex.h.

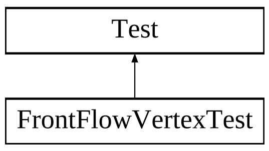
The documentation for this struct was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow_vertex.h

6.24 FrontFlowVertexTest Class Reference

`#include <front_flow_vertex_test.h>`

Inheritance diagram for FrontFlowVertexTest:



Public Types

- typedef FrontFlowVertex< int > VertexType
- typedef ListNode< VertexType > NodeType
- typedef List< NodeType > ListType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< VertexType > _default_vertex
- std::shared_ptr< VertexType > _normal_vertex
- std::shared_ptr< ListType > _list
- std::shared_ptr< NodeType > _nodes [FFV_NUM]

6.24.1 Detailed Description

Definition at line 30 of file front_flow_vertex_test.h.

6.24.2 Member Typedef Documentation

6.24.2.1 typedef **List<NodeType> FrontFlowVertexTest::ListType**

Definition at line 35 of file front_flow_vertex_test.h.

6.24.2.2 typedef **ListNode<VertexType> FrontFlowVertexTest::NodeType**

Node

Definition at line 34 of file front_flow_vertex_test.h.

6.24.2.3 typedef **FrontFlowVertex<int> FrontFlowVertexTest::VertexType**

Definition at line 33 of file front_flow_vertex_test.h.

6.24.3 Member Function Documentation

6.24.3.1 void FrontFlowVertexTest::SetUp ( ) `[inline]`, `[protected]`

Definition at line 38 of file front_flow_vertex_test.h.

6.24.3.2 void FrontFlowVertexTest::TearDown ( ) `[inline]`, `[protected]`

Definition at line 49 of file front_flow_vertex_test.h.

6.24.4 Member Data Documentation

6.24.4.1 std::shared_ptr<**VertexType**> FrontFlowVertexTest::_default_vertex `[protected]`

Definition at line 51 of file front_flow_vertex_test.h.

6.24.4.2 std::shared_ptr< ListType > FrontFlowVertexTest::_list [protected]

Definition at line 54 of file front_flow_vertex_test.h.

6.24.4.3 std::shared_ptr< NodeType > FrontFlowVertexTest::_nodes[FFV_NUM] [protected]

Definition at line 55 of file front_flow_vertex_test.h.

6.24.4.4 std::shared_ptr< VertexType > FrontFlowVertexTest::_normal_vertex [protected]

Definition at line 52 of file front_flow_vertex_test.h.

The documentation for this class was generated from the following file:

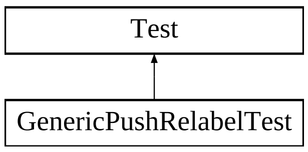
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow_vertex_test.h

6.25 GenericPushRelabelTest Class Reference

GenericPushRelabelTest:

`#include <genericpushrelabel_test.h>`

Inheritance diagram for GenericPushRelabelTest:



Public Types

- typedef Graph< PR_N, FlowVertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.25.1 Detailed Description

GenericPushRelabelTest:

`GenericPushRelabelTest ::testing::Test TEST_F`

Definition at line 43 of file genericpushrelabel_test.h.

6.25.2 Member Typedef Documentation

6.25.2.1 typedef **Graph<PR_N,FlowVertex<int> > GenericPushRelabelTest::GType**

`FlowVertex<int>`

Definition at line 46 of file genericpushrelabel_test.h.

6.25.3 Member Function Documentation

6.25.3.1 void GenericPushRelabelTest::SetUp ( ) `[inline], [protected]`

Definition at line 49 of file genericpushrelabel_test.h.

6.25.3.2 void GenericPushRelabelTest::TearDown ( ) `[inline], [protected]`

Definition at line 67 of file genericpushrelabel_test.h.

6.25.4 Member Data Documentation

6.25.4.1 std::shared_ptr<**GType**> GenericPushRelabelTest::_graph `[protected]`

26-6

Definition at line 68 of file genericpushrelabel_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/max_flow/generic_push_relabel/genericpushrelabel_test.h

6.26 IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType > Struct Template Reference

Graph2222.1

`#include <graph.h>`

Public Types

- typedef int VIDType
- typedef int EWeightType
- typedef std::tuple< VIDType, VIDType, EWeightType > EdgeTupleType
- typedef VType VertexType

Public Member Functions

- Graph (EWeightType val)

    invalid_weight
- Graph ()

- VIDType add_vertex (const typename VertexType::KeyType &key)

    add_vertex:

- VIDType add_vertex (const typename VertexType::KeyType &key, VIDType id)

  add_vertex:
- void modify_vertex (const typename VertexType::KeyType &newkey, VIDType id)

  modify_vertex:
- void add_edge (const EdgeTupleType &edge_tuple)

  add_edge:
- template<typename Iterator >
  void add_edges (const Iterator &begin, const Iterator &end)

  add_edges:
- void adjust_edge (VIDType id1, VIDType id2, EWeightType wt)

  adjust_edge:
- const std::vector< EdgeTupleType > edge_tuples () const

  edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`
- const std::vector< EdgeTupleType > vertex_edge_tuples (VIDType id) const

  vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`
- bool has_edge (VIDType id_from, VIDType id_to) const

  has_edge:
- EWeightType weight (VIDType id_from, VIDType id_to) const

  weight:
- std::shared_ptr< Graph > inverse ()

  inverse:

Public Attributes

- std::array< std::shared_ptr< VertexType >, N > vertexes
- std::size_t next_empty_vertex
- MatrixGraph< N > matrix
- ADJListGraph< N > adjList

Static Public Attributes

- static const unsigned NUM =N

6.26.1 Detailed Description

template<unsigned N, typename VType>struct IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >

Graph2222.1

- `matrixMatrixGraph<N>`
- `adjListADJListGraph<N>`
- `vertexesstd::array<std::shared_ptr<VertexType>,N>std::array`
- `next_empty_vertexstd::size_t`

Definition at line 42 of file graph.h.

6.26.2 Member Typedef Documentation

6.26.2.1 template<unsigned N, typename VType > typedef std::tuple<VIDType,VIDType,EWeightType> IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::EdgeTupleType

12)

Definition at line 46 of file graph.h.

6.26.2.2 template<unsigned N, typename VType > typedef int IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::EWeightType

Definition at line 45 of file graph.h.

6.26.2.3 template<unsigned N, typename VType > typedef VType IntroductionToAlgorithm::GraphAlgorithm::↩ Graph< N, VType >::VertexType

Definition at line 47 of file graph.h.

6.26.2.4 template<unsigned N, typename VType > typedef int IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::VIDType

Definition at line 44 of file graph.h.

6.26.3 Constructor & Destructor Documentation

6.26.3.1 template<unsigned N, typename VType > IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::Graph (EWeightType *val*) [inline], [explicit]

`invalid_weight`

Parameters

val:	

Definition at line 54 of file graph.h.

6.26.3.2 template<unsigned N, typename VType > IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::Graph ( ) [inline]

Definition at line 57 of file graph.h.

6.26.4 Member Function Documentation

6.26.4.1 template<unsigned N, typename VType > void IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::add_edge (const EdgeTupleType & *edge_tuple*) [inline]

add_edge:

Parameters

edge_tuple:	Edge`std::tuple<VIDType,VIDType,EWeightType>`

- `id[0,N)`
- `id`

Definition at line 129 of file graph.h.

6.26.4.2 template< unsigned N, typename VType > template< typename Iterator > void IntroductionToAlgorithm↩ ::GraphAlgorithm::Graph< N, VType >::add_edges (const Iterator & *begin*, const Iterator & *end*) `[inline]`

add_edges:

Parameters

begin:	
end:	Edge`std::tuple<VIDType,VIDType,EWeightType>`

Definition at line 151 of file graph.h.

6.26.4.3 template< unsigned N, typename VType > VIDType IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::add_vertex (const typename VertexType::KeyType & *key*) `[inline]`

add_vertex:

Parameters

key:	

Returns

: id

N`std::invalid_argument`.

`next_empty_vertex`ůstd::invalid_argument‘

Definition at line 68 of file graph.h.

6.26.4.4 template< unsigned N, typename VType > VIDType IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::add_vertex (const typename VertexType::KeyType & *key*, VIDType *id*) `[inline]`

add_vertex:

Parameters

key:	
id:<tt>id</tt>	

Returns

: id

- `id<0id>=Nid[0,N)`
- `idid`

Definition at line 88 of file graph.h.

6.26.4.5 template<unsigned N, typename VType > void IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::adjust_edge (VIDType *id1*, VIDType *id2*, EWeightType *wt*) `[inline]`

adjust_edge:

Parameters

id1:	
id2:	
wt:	

- `id[0,N)`
- `id`

Definition at line 174 of file graph.h.

6.26.4.6 template<unsigned N, typename VType > const std::vector<EdgeTupleType> IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::edge_tuples ( ) const `[inline]`

edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Returns

:

Definition at line 189 of file graph.h.

6.26.4.7 template<unsigned N, typename VType > bool IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::has_edge (VIDType *id_from*, VIDType *id_to*) const `[inline]`

has_edge:

Parameters

id_from	`id`
id_to	`id`

Returns

:

- `id[0,N)`
- `id`

Definition at line 253 of file graph.h.

6.26.4.8 template< unsigned N, typename VType > std::shared_ptr< Graph > IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::inverse ( ) [inline]

inverse:

Returns

:

-
-

`inverse`

Definition at line 299 of file graph.h.

6.26.4.9 template< unsigned N, typename VType > void IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::modify_vertex (const typename VertexType::KeyType & *newkey,* VIDType *id*) [inline]

modify_vertex:

Parameters

newkey:	
id:<tt>id</tt>	• `id<0id>=Nid[0,N)` • `idid`

Definition at line 105 of file graph.h.

6.26.4.10 template< unsigned N, typename VType > const std::vector< EdgeTupleType > IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::vertex_edge_tuples (VIDType *id*) const [inline]

vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Parameters

id	`id`

Returns

:

- `id[0,N)`
- `id`

Definition at line 217 of file graph.h.

6.26.4.11 template< unsigned N, typename VType > EWeightType IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::weight (VIDType *id_from,* VIDType *id_to*) const [inline]

weight:

Parameters

id_from	`id`
id_to	`id`

Returns

:

- `id[0,N)`
- `id`

Definition at line 278 of file graph.h.

6.26.5 Member Data Documentation

6.26.5.1 template<unsigned N, typename VType > ADJListGraph<N> IntroductionToAlgorithm::Graph↩ Algorithm::Graph< N, VType >::adjList

Definition at line 319 of file graph.h.

6.26.5.2 template<unsigned N, typename VType > MatrixGraph<N> IntroductionToAlgorithm::GraphAlgorithm↩ ::Graph< N, VType >::matrix

Definition at line 318 of file graph.h.

6.26.5.3 template<unsigned N, typename VType > std::size_t IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::next_empty_vertex

Definition at line 317 of file graph.h.

6.26.5.4 template<unsigned N, typename VType > const unsigned IntroductionToAlgorithm::GraphAlgorithm::↩ Graph< N, VType >::NUM =N `[static]`

Definition at line 48 of file graph.h.

6.26.5.5 template<unsigned N, typename VType > std::array<std::shared_ptr<VertexType>,N> IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >::vertexes

Definition at line 316 of file graph.h.

The documentation for this struct was generated from the following file:

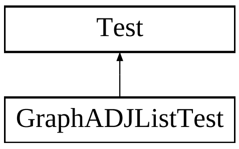
- src/graph_algorithms/basic_graph/graph_representation/graph/graph.h

6.27 GraphADJListTest Class Reference

GraphADJListTest:

`#include <adjlistgraph_test.h>`

Inheritance diagram for GraphADJListTest:



Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< ADJListGraph< ADJ_NUM > > graph

6.27.1 Detailed Description

GraphADJListTest:

`GraphADJListTest ::testing::Test TEST_F`

Definition at line 34 of file adjlistgraph_test.h.

6.27.2 Member Function Documentation

6.27.2.1 void GraphADJListTest::SetUp ( ) `[inline], [protected]`

Definition at line 39 of file adjlistgraph_test.h.

6.27.2.2 void GraphADJListTest::TearDown ( ) `[inline], [protected]`

Definition at line 42 of file adjlistgraph_test.h.

6.27.3 Member Data Documentation

6.27.3.1 std::shared_ptr<ADJListGraph<ADJ_NUM> > GraphADJListTest::graph `[protected]`

Definition at line 44 of file adjlistgraph_test.h.

The documentation for this class was generated from the following file:

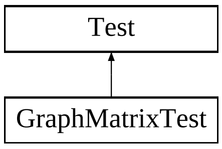
- src/graph_algorithms/basic_graph/graph_representation/adjlist_graph/adjlistgraph_test.h

6.28 GraphMatrixTest Class Reference

GraphMatrixTest:

`#include <matrixgraph_test.h>`

Inheritance diagram for GraphMatrixTest:



Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< MatrixGraph< MTXNUM > > graph

6.28.1 Detailed Description

GraphMatrixTest:

```
GraphMatrixTest ::testing::Test TEST_F
```

Definition at line 34 of file matrixgraph_test.h.

6.28.2 Member Function Documentation

6.28.2.1 void GraphMatrixTest::SetUp ( ) [inline], [protected]

Definition at line 37 of file matrixgraph_test.h.

6.28.2.2 void GraphMatrixTest::TearDown ( ) [inline], [protected]

Definition at line 40 of file matrixgraph_test.h.

6.28.3 Member Data Documentation

6.28.3.1 std::shared_ptr<**MatrixGraph**<MTXNUM> > GraphMatrixTest::graph [protected]

Definition at line 42 of file matrixgraph_test.h.

The documentation for this class was generated from the following file:

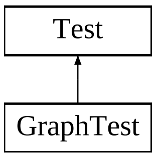
- src/graph_algorithms/basic_graph/graph_representation/matrix_graph/matrixgraph_test.h

6.29 GraphTest Class Reference

GraphTest:

```
#include <graph_test.h>
```

Inheritance diagram for GraphTest:



Public Types

- typedef Graph< G_N, Vertex< double > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _empty_graph
- std::shared_ptr< GType > _n_vertexes_graph
- std::shared_ptr< GType > _n_vertexes_m_edges_graph

6.29.1 Detailed Description

GraphTest:

`GraphTest ::testing::Test TEST_F`

Definition at line 36 of file graph_test.h.

6.29.2 Member Typedef Documentation

6.29.2.1 typedef **Graph**<G_N,**Vertex**<double> > **GraphTest::GType**

`Vertex<double>`

Definition at line 39 of file graph_test.h.

6.29.3 Member Function Documentation

6.29.3.1 void GraphTest::SetUp ( ) [inline], [protected]

Definition at line 42 of file graph_test.h.

6.29.3.2 void GraphTest::TearDown ( ) [inline], [protected]

Definition at line 58 of file graph_test.h.

6.29.4 Member Data Documentation

6.29.4.1 std::shared_ptr<**GType**> GraphTest::_empty_graph [protected]

Definition at line 60 of file graph_test.h.

6.29.4.2 std::shared_ptr< GType > GraphTest::_n_vertexes_graph [protected]

n,0

Definition at line 61 of file graph_test.h.

6.29.4.3 std::shared_ptr< GType > GraphTest::_n_vertexes_m_edges_graph [protected]

n,m

Definition at line 62 of file graph_test.h.

The documentation for this class was generated from the following file:

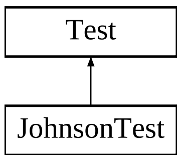
- src/graph_algorithms/basic_graph/graph_representation/graph/graph_test.h

6.30 JohnsonTest Class Reference

JohnsonTest:

`#include <johnson_test.h>`

Inheritance diagram for JohnsonTest:



Public Types

- typedef Graph< JS_N, VertexP< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.30.1 Detailed Description

JohnsonTest:

`JohnsonTest ::testing::Test TEST_F`

Definition at line 40 of file johnson_test.h.

6.30.2 Member Typedef Documentation

6.30.2.1 typedef **Graph<JS_N,VertexP<int> > JohnsonTest::GType**

`VertexP<int>`

Definition at line 43 of file johnson_test.h.

6.30.3 Member Function Documentation

6.30.3.1 void JohnsonTest::SetUp ( ) [inline], [protected]

Definition at line 46 of file johnson_test.h.

6.30.3.2 void JohnsonTest::TearDown ( ) [inline], [protected]

Definition at line 64 of file johnson_test.h.

6.30.4 Member Data Documentation

6.30.4.1 std::shared_ptr<**GType**> JohnsonTest::_graph [protected]

25-1

Definition at line 65 of file johnson_test.h.

The documentation for this class was generated from the following file:

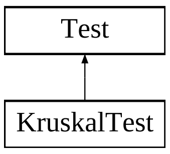
- src/graph_algorithms/all_node_pair_shortest_path/johnson/johnson_test.h

6.31 KruskalTest Class Reference

KruskalTest:

`#include <kruskal_test.h>`

Inheritance diagram for KruskalTest:



Public Types

- typedef Graph< K_NUM, SetVertex< int > > GType
- typedef std::function< void(SetVertex< int >::VIDType,SetVertex< int >::VIDType)> ActionType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _list_graph
- std::shared_ptr< GType > _all_edges_graph

6.31.1 Detailed Description

KruskalTest:

`KruskalTest ::testing::Test TEST_F`

Definition at line 38 of file kruskal_test.h.

6.31.2 Member Typedef Documentation

6.31.2.1 typedef std::function<void(SetVertex<int>::VIDType ,SetVertex<int>::VIDType)> KruskalTest::ActionType

Action

Definition at line 42 of file kruskal_test.h.

6.31.2.2 typedef Graph<K_NUM,SetVertex<int> > KruskalTest::GType

`SetVertex<int>`

Definition at line 41 of file kruskal_test.h.

6.31.3 Member Function Documentation

6.31.3.1 void KruskalTest::SetUp ( ) `[inline]`, `[protected]`

Definition at line 44 of file kruskal_test.h.

6.31.3.2 void KruskalTest::TearDown ( ) `[inline]`, `[protected]`

Definition at line 72 of file kruskal_test.h.

6.31.4 Member Data Documentation

6.31.4.1 std::shared_ptr<GType> KruskalTest::_1e_graph `[protected]`

Definition at line 74 of file kruskal_test.h.

6.31.4.2 std::shared_ptr<GType> KruskalTest::_1v_graph `[protected]`

Definition at line 73 of file kruskal_test.h.

6.31.4.3 std::shared_ptr<GType> KruskalTest::_all_edges_graph `[protected]`

Definition at line 76 of file kruskal_test.h.

6.31.4.4 std::shared_ptr< GType > KruskalTest::_list_graph [protected]

Definition at line 75 of file kruskal_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/minimum_spanning_tree/kruskal/kruskal_test.h

6.32 IntroductionToAlgorithm::GraphAlgorithm::List< NodeType > Struct Template Reference

List

`#include <front_flow_vertex.h>`

Public Member Functions

- List ()

  List
- void add (std::shared_ptr< NodeType > element)

  add:
- std::shared_ptr< NodeType > front_of (std::shared_ptr< NodeType > element) const

  front_of:
- std::string to_string () const

  to_string

Public Attributes

- std::shared_ptr< NodeType > head
- std::shared_ptr< NodeType > current

6.32.1 Detailed Description

template<typename NodeType>struct IntroductionToAlgorithm::GraphAlgorithm::List< NodeType >

List

- head
- current:

Definition at line 37 of file front_flow_vertex.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 template<typename NodeType> IntroductionToAlgorithm::GraphAlgorithm::List< NodeType >::List ( ) [inline]

List

headcurrent

Definition at line 48 of file front_flow_vertex.h.

6.32.3 Member Function Documentation

6.32.3.1 template<typename NodeType> void IntroductionToAlgorithm::GraphAlgorithm::List< NodeType >::add (std::shared_ptr< NodeType > *element*) [inline]

add:

Parameters

element:	elementelement

Definition at line 58 of file front_flow_vertex.h.

6.32.3.2 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::↩ GraphAlgorithm::List< NodeType >::front_of (std::shared_ptr< NodeType > *element*) const [inline]

front_of:

Parameters

element:	

Returns

:

element

Definition at line 81 of file front_flow_vertex.h.

6.32.3.3 template<typename NodeType> std::string IntroductionToAlgorithm::GraphAlgorithm::List< NodeType >::to_string ( ) const [inline]

to_string

Returns

:

Definition at line 106 of file front_flow_vertex.h.

6.32.4 Member Data Documentation

6.32.4.1 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::GraphAlgorithm↩ ::List< NodeType >::current

Definition at line 40 of file front_flow_vertex.h.

6.32.4.2 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::GraphAlgorithm↩ ::List< NodeType >::head

Definition at line 39 of file front_flow_vertex.h.

The documentation for this struct was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow_vertex.h

6.33 IntroductionToAlgorithm::GraphAlgorithm::ListNode< ValueType > Struct Template Reference

ListNode

`#include <front_flow_vertex.h>`

Public Member Functions

- ListNode ()

  ListNode
- std::string to_string () const

  to_string

Public Attributes

- std::weak_ptr< ValueType > value
- std::shared_ptr< ListNode > next

6.33.1 Detailed Description

template<typename ValueType>struct IntroductionToAlgorithm::GraphAlgorithm::ListNode< ValueType >

ListNode

- value
- next:

Definition at line 133 of file front_flow_vertex.h.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 template<typename ValueType > IntroductionToAlgorithm::GraphAlgorithm::ListNode< ValueType >::ListNode ( ) [inline]

ListNode

valuenext

Definition at line 141 of file front_flow_vertex.h.

6.33.3 Member Function Documentation

6.33.3.1 template<typename ValueType > std::string IntroductionToAlgorithm::GraphAlgorithm::ListNode< ValueType >::to_string ( ) const [inline]

to_string

Returns

:

`value`

Definition at line 150 of file front_flow_vertex.h.

6.33.4 Member Data Documentation

6.33.4.1 template<typename ValueType > std::shared_ptr<ListNode> IntroductionToAlgorithm::Graph↩ Algorithm::ListNode< ValueType >::next

Definition at line 143 of file front_flow_vertex.h.

6.33.4.2 template<typename ValueType > std::weak_ptr<ValueType> IntroductionToAlgorithm::GraphAlgorithm::↩ ListNode< ValueType >::value

Definition at line 142 of file front_flow_vertex.h.

The documentation for this struct was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow_vertex.h

6.34 IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N > Struct Template Reference

MatrixGraph2222.1

`#include <matrixgraph.h>`

Public Types

- typedef int VIDType
- typedef int EWeightType
- typedef std::tuple< VIDType, VIDType, EWeightType > EdgeTupleType

Public Member Functions

- MatrixGraph (EWeightType val)

  `invalid_weight`
- MatrixGraph ()

  `invalid_weight0`
- void add_edge (const EdgeTupleType &edge_tuple)

  add_edge:
- template<typename Iteator >
  void add_edges (const Iteator &begin, const Iteator &end)

  add_edges:
- void adjust_edge (VIDType id1, VIDType id2, EWeightType wt)

  adjust_edge:
- const std::vector< EdgeTupleType > edge_tuples () const

  edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

- const std::vector< EdgeTupleType > vertex_edge_tuples (VIDType id) const

    vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`
- bool has_edge (VIDType id_from, VIDType id_to) const

    has_edge:
- EWeightType weight (VIDType id_from, VIDType id_to) const

    weight:

Public Attributes

- std::array< std::array< EWeightType, N >, N > matrix
- const EWeightType invalid_weight

Static Public Attributes

- static const unsigned NUM =N

6.34.1 Detailed Description

template<unsigned N>struct IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >

MatrixGraph2222.1

- `matrixstd::array<std::array<EWeightType,N>, N>N*N`
- `invalid_weightrcrc`

Definition at line 37 of file matrixgraph.h.

6.34.2 Member Typedef Documentation

6.34.2.1 template<unsigned N> typedef std::tuple<VIDType,VIDType,EWeightType> IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::EdgeTupleType

12)

Definition at line 41 of file matrixgraph.h.

6.34.2.2 template<unsigned N> typedef int IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::EWeightType

Definition at line 40 of file matrixgraph.h.

6.34.2.3 template<unsigned N> typedef int IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::VIDType

Definition at line 39 of file matrixgraph.h.

6.34.3 Constructor & Destructor Documentation

6.34.3.1 template<unsigned N> IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::MatrixGraph (EWeightType *val*) [inline], [explicit]

`invalid_weight`

Parameters

val:	

Definition at line 47 of file matrixgraph.h.

6.34.3.2 template<unsigned N> IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::MatrixGraph ( ) [inline]

`invalid_weight`0

Definition at line 54 of file matrixgraph.h.

6.34.4 Member Function Documentation

6.34.4.1 template<unsigned N> void IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::add_edge (const EdgeTupleType & *edge_tuple*) [inline]

add_edge:

Parameters

edge_tuple:	`Edgestd::tuple<VIDType,VIDType,EWeightType>`

`std::invalid_argument`

`[0,N)id`

Definition at line 66 of file matrixgraph.h.

6.34.4.2 template<unsigned N> template<typename Iteator > void IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::add_edges (const Iteator & *begin,* const Iteator & *end*) [inline]

add_edges:

Parameters

begin:	
end:	`Edgestd::tuple<VIDType,VIDType,EWeightType>`

`std::invalid_argument`

`[0,N)id`

Definition at line 87 of file matrixgraph.h.

6.34.4.3 template<unsigned N> void IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::adjust_edge (VIDType *id1,* VIDType *id2,* EWeightType *wt*) [inline]

adjust_edge:

Parameters

id1:	
id2:	

wt:	`id1id2wtstd::invalid_argument`
	`id1id2[0,N)id`

Definition at line 108 of file matrixgraph.h.

6.34.4.4 template<unsigned N> const std::vector<EdgeTupleType> IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::edge_tuples ( ) const [inline]

edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Returns

:

Definition at line 121 of file matrixgraph.h.

6.34.4.5 template<unsigned N> bool IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::has_edge (VIDType *id_from,* VIDType *id_to*) const [inline]

has_edge:

Parameters

id_from	`id`
id_to	`id`

Returns

:

- `id_fromid_to >id_fromid_to[0,N)`
- `id_fromid_totrue`
- `id_fromid_tofalse`

Definition at line 168 of file matrixgraph.h.

6.34.4.6 template<unsigned N> const std::vector<EdgeTupleType> IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::vertex_edge_tuples (VIDType *id*) const [inline]

vertex_edge_tuples:`std::vector<std::tuple<VIDType,VIDType,EWeightType>>`

Parameters

id	`id`

Returns

:

- `id[0,N)`

Definition at line 141 of file matrixgraph.h.

6.34.4.7 template< unsigned N > EWeightType IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::weight (VIDType *id_from,* VIDType *id_to*) const [inline]

weight:

Parameters

id_from	`id`
id_to	`id`

Returns

:

`id_fromid_tostd::invalid_argument`

- `id_fromid_to >id_fromid_to[0,N)`
- `id_fromid_to`

Definition at line 189 of file matrixgraph.h.

6.34.5 Member Data Documentation

6.34.5.1 template<unsigned N> const EWeightType IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::invalid_weight

const

Definition at line 201 of file matrixgraph.h.

6.34.5.2 template<unsigned N> std::array<std::array<EWeightType,N>, N> IntroductionToAlgorithm::Graph↩ Algorithm::MatrixGraph< N >::matrix

Definition at line 200 of file matrixgraph.h.

6.34.5.3 template<unsigned N> const unsigned IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >::NUM =N `[static]`

Definition at line 42 of file matrixgraph.h.

The documentation for this struct was generated from the following file:

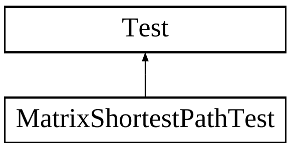
- src/graph_algorithms/basic_graph/graph_representation/matrix_graph/matrixgraph.h

6.35 MatrixShortestPathTest Class Reference

MatrixShortestPathTest:

`#include <matrix_shortest_path_test.h>`

Inheritance diagram for MatrixShortestPathTest:



Public Types

- typedef Graph< MT_N, Vertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.35.1 Detailed Description

MatrixShortestPathTest:

`MatrixShortestPathTest ::testing::Test TEST_F`

Definition at line 42 of file matrix_shortest_path_test.h.

6.35.2 Member Typedef Documentation

6.35.2.1 typedef **Graph**<MT_N,**Vertex**<int> > **MatrixShortestPathTest::GType**

`Vertex<int>`

Definition at line 45 of file matrix_shortest_path_test.h.

6.35.3 Member Function Documentation

6.35.3.1 void MatrixShortestPathTest::SetUp ( ) `[inline], [protected]`

Definition at line 48 of file matrix_shortest_path_test.h.

6.35.3.2 void MatrixShortestPathTest::TearDown ( ) `[inline], [protected]`

Definition at line 66 of file matrix_shortest_path_test.h.

6.35.4 Member Data Documentation

6.35.4.1 std::shared_ptr<**GType**> MatrixShortestPathTest::_graph `[protected]`

25-1

Definition at line 67 of file matrix_shortest_path_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/all_node_pair_shortest_path/matrix_shortest_path/matrix_shortest_path_test.h

6.36 IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType > Class Template Reference

MinQueue66.5

```
#include <minqueue.h>
```

Public Types

- typedef std::function< bool(std::shared_ptr< T >, std::shared_ptr< T >)> CompareType
- typedef std::function< TKeyType &(std::shared_ptr< T >)> GetKeyType

Public Member Functions

- MinQueue (CompareType compare, GetKeyType getKey)
- MinQueue (std::size_t reseve_size, CompareType compare, GetKeyType getKey)
- std::shared_ptr< T > min ()

  min:
- std::shared_ptr< T > extract_min ()

  extract_min:
- int insert (std::shared_ptr< T > element)

  insert:
- bool is_empty ()

  is_empty:
- int index_inqueue (std::shared_ptr< T > element)

  is_inqueue:
- void decreate_key (std::size_t element_index, TKeyType new_key)

  decreate_key:key
- void setupHeap ()

  setupHeap:
- void heapify (std::size_t elementIndex)

  heapify

Protected Member Functions

- std::size_t _parentIndex (std::size_t elementIndex, bool &valid)

  _parentIndex:
- std::size_t _lchildIndex (std::size_t elementIndex, bool &valid)

  _lchildIndex:
- std::size_t _rchildIndex (std::size_t elementIndex, bool &valid)

  _rchildIndex:

Private Attributes

- std::vector< std::shared_ptr< T > > _data
- std::size_t _size
- CompareType _compare
- GetKeyType _getKey

6.36.1 Detailed Description

template<typename T, typename TKeyType>class IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >

MinQueue66.5

S

- insert(S,x):xS
- min(S):S
- extract_min(S):S
- decrease_key(S,x,k):xk,kx

-
  - is_empty(S): S
  - is_inqueue(S,x):xS
  - setupHeap(S):)
  - heapify(S,index):
- `_data_data`0`reseve_size_data` 1/4`extract_min`
- classstructint,double
  - _getKey:std::function<TKeyType&(std::shared_ptr<T>)>std::shared_ptr<T>TKeyType TTKeyType↩ T_getKey
    - classTKeyTypeT_getKey
    - intTKeyTypeT_getKey
  - _compare:std::function<bool (std::shared_ptr<T>,std::shared_ptr<T>)>

Definition at line 60 of file minqueue.h.

6.36.2 Member Typedef Documentation

6.36.2.1 template<typename T, typename TKeyType> typedef std::function<bool (std::shared_ptr<T>,std↩ ::shared_ptr<T>)> IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::CompareType

std::shared_ptr<T>

Definition at line 63 of file minqueue.h.

6.36.2.2 template<typename T, typename TKeyType> typedef std::function<TKeyType&(std::shared_ptr<T>)> IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::GetKeyType

std::shared_ptr<T>

Definition at line 64 of file minqueue.h.

6.36.3 Constructor & Destructor Documentation

6.36.3.1 template<typename T, typename TKeyType> IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::MinQueue (CompareType *compare,* GetKeyType *getKey*) `[inline]`

Parameters

compare:std↩ ::shared_ptr<↩ T>	
getKeystd↩ ::shared_ptr<↩ T> TKey&Tkey↩ Tkey	

Definition at line 71 of file minqueue.h.

6.36.3.2 template<typename T, typename TKeyType> IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::MinQueue (std::size_t *reseve_size,* CompareType *compare,* GetKeyType *getKey*) [inline]

Parameters

reseve_size:	
compare:std↩ ::shared_ptr<↩ T>	
getKeystd↩ ::shared_ptr<↩ T> TKey&Tkey↩ Tkey	

Definition at line 79 of file minqueue.h.

6.36.4 Member Function Documentation

6.36.4.1 template<typename T, typename TKeyType> std::size_t IntroductionToAlgorithm::QueueAlgorithm↩ ::MinQueue< T, TKeyType >::_lchildIndex (std::size_t *elementIndex,* bool & *valid*) [inline], [protected]

_lchildIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex/2)+1

- 01
-

Definition at line 326 of file minqueue.h.

6.36.4.2 template<typename T, typename TKeyType> std::size_t IntroductionToAlgorithm::QueueAlgorithm↩ ::MinQueue< T, TKeyType >::_parentIndex (std::size_t *elementIndex,* bool & *valid*) [inline], [protected]

_parentIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex-1)/2

-

Definition at line 303 of file minqueue.h.

6.36.4.3 template< typename T, typename TKeyType > std::size_t IntroductionToAlgorithm::QueueAlgorithm↩ ::MinQueue< T, TKeyType >::_rchildIndex (std::size_t *elementIndex,* bool & *valid*) [inline], [protected]

_rchildIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex/2)+2

- 012
-

Definition at line 353 of file minqueue.h.

6.36.4.4 template< typename T, typename TKeyType > void IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::decreate_key (std::size_t *element_index,* TKeyType *new_key*) [inline]

decreate_key:`key`

Parameters

element_index	
new_↩ key<tt>key</tt>↩ TKey	

- `element_index`
- `new_keykey`

- `key`

-
    -
    -
    -
- O(h)
-

Definition at line 216 of file minqueue.h.

6.36.4.5 template<typename T, typename TKeyType> std::shared_ptr<T> IntroductionToAlgorithm::Queue↩ Algorithm::MinQueue< T, TKeyType >::extract_min ( ) [inline]

extract_min:

Returns

-
-
    -
    - `_size`
    - `heapify(0)`
    -

1/4_size=02

- O(h),h
-

Definition at line 116 of file minqueue.h.

6.36.4.6 template<typename T, typename TKeyType> void IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::heapify (std::size_t *elementIndex*) [inline]

heapify

Parameters

elementIndex	:

Returns

void

heapify

- O(n)
-

Definition at line 265 of file minqueue.h.

6.36.4.7 template<typename T, typename TKeyType> int IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::index_inqueue (std::shared_ptr< T > *element*) [inline]

is_inqueue:

Parameters

element:	

Returns

`element` std::vector-1

- O(h)

Definition at line 185 of file minqueue.h.

6.36.4.8 template<typename T, typename TKeyType> int IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::insert (std::shared_ptr< T > *element*) `[inline]`

insert:

Parameters

element	

Returns

: -1

`_data_size*2+2`

- `_size==_data.size()`

- `keykey`
- `decreate_key(..)`

_size=02

- O(h)
-

Definition at line 148 of file minqueue.h.

6.36.4.9 template<typename T, typename TKeyType> bool IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::is_empty ( ) `[inline]`

is_empty:

Returns

`_size`0

- O(1)

Definition at line 170 of file minqueue.h.

6.36.4.10 template<typename T, typename TKeyType> std::shared_ptr<T> IntroductionToAlgorithm::Queue↩ Algorithm::MinQueue< T, TKeyType >::min ( ) [inline]

min:

Returns

- O(1)

Definition at line 92 of file minqueue.h.

6.36.4.11 template<typename T, typename TKeyType> void IntroductionToAlgorithm::QueueAlgorithm::Min↩ Queue< T, TKeyType >::setupHeap ( ) [inline]

setupHeap:

Returns

void

heapify

- O(nlogn)
-

Definition at line 243 of file minqueue.h.

6.36.5 Member Data Documentation

6.36.5.1 template<typename T, typename TKeyType> CompareType IntroductionToAlgorithm::QueueAlgorithm↩ ::MinQueue< T, TKeyType >::_compare [private]

std::shared_ptr<T>

Definition at line 371 of file minqueue.h.

6.36.5.2 template<typename T, typename TKeyType> std::vector<std::shared_ptr<T> > IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >::_data [private]

Definition at line 369 of file minqueue.h.

6.36.5.3 template<typename T, typename TKeyType> GetKeyType IntroductionToAlgorithm::QueueAlgorithm::↩ MinQueue< T, TKeyType >::_getKey [private]

std::shared_ptr<T>

Definition at line 372 of file minqueue.h.

6.36.5.4 template<typename T, typename TKeyType> std::size_t IntroductionToAlgorithm::QueueAlgorithm::Min↩ Queue< T, TKeyType >::_size [private]

Definition at line 370 of file minqueue.h.

The documentation for this class was generated from the following file:

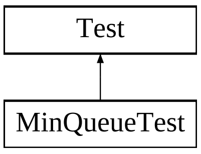
- src/queue_algorithms/min_queue/minqueue.h

6.37 MinQueueTest Class Reference

MinQueueTest:

```
#include <minqueue_test.h>
```

Inheritance diagram for MinQueueTest:



Public Types

- typedef std::function< bool(std::shared_ptr< int >, std::shared_ptr< int >)> Int_Compare_Type
- typedef std::function< int &(std::shared_ptr< int >)> Int_Get_Type
- typedef std::function< bool(std::shared_ptr< Node >, std::shared_ptr< Node >)> Struct_Compare_Type
- typedef std::function< double &(std::shared_ptr< Node >)> Struct_Get_Type
- typedef MinQueue< Node, double > Struct_MinQueue_Type
- typedef MinQueue< int, int > Int_MinQueue_Type

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< Struct_MinQueue_Type > _strcut_minqueue
- std::shared_ptr< Int_MinQueue_Type > _int_minqueue

6.37.1 Detailed Description

MinQueueTest:

```
MinQueueTest ::testing::Test TEST_F
```

Definition at line 42 of file minqueue_test.h.

6.37.2 Member Typedef Documentation

6.37.2.1 typedef std::function<bool (std::shared_ptr<int>,std::shared_ptr<int>)> **MinQueueTest::Int_Compare_↩ Type**

int∗

Definition at line 45 of file minqueue_test.h.

6.37.2.2 typedef std::function<int&(std::shared_ptr<int>)> **MinQueueTest::Int_Get_Type**

int∗key

Definition at line 46 of file minqueue_test.h.

6.37.2.3 typedef **MinQueue**<int,int> **MinQueueTest::Int_MinQueue_Type**

Node∗Nodedouble

Definition at line 50 of file minqueue_test.h.

6.37.2.4 typedef std::function<bool (std::shared_ptr<**Node**>,std::shared_ptr<**Node**>)> **MinQueueTest::Struct_Compare_Type**

Node∗

Definition at line 47 of file minqueue_test.h.

6.37.2.5 typedef std::function<double&(std::shared_ptr<**Node**>)> **MinQueueTest::Struct_Get_Type**

Node∗key

Definition at line 48 of file minqueue_test.h.

6.37.2.6 typedef **MinQueue**<**Node**,double> **MinQueueTest::Struct_MinQueue_Type**

int∗int

Definition at line 49 of file minqueue_test.h.

6.37.3 Member Function Documentation

6.37.3.1 void MinQueueTest::SetUp ( ) `[inline]`, `[protected]`

Definition at line 52 of file minqueue_test.h.

6.37.3.2 void MinQueueTest::TearDown ( ) `[inline]`, `[protected]`

Definition at line 62 of file minqueue_test.h.

6.37.4 Member Data Documentation

6.37.4.1 std::shared_ptr<Int_MinQueue_Type> MinQueueTest::_int_minqueue [protected]

Definition at line 64 of file minqueue_test.h.

6.37.4.2 std::shared_ptr<Struct_MinQueue_Type> MinQueueTest::_strcut_minqueue [protected]

Definition at line 63 of file minqueue_test.h.

The documentation for this class was generated from the following file:

- src/queue_algorithms/min_queue/minqueue_test.h

6.38 Node Struct Reference

Node:

`#include <minqueue_test.h>`

Public Member Functions

- Node (int k)

Public Attributes

- double key

6.38.1 Detailed Description

Node:

Definition at line 31 of file minqueue_test.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Node::Node (int *k*) [inline]

Definition at line 33 of file minqueue_test.h.

6.38.3 Member Data Documentation

6.38.3.1 double Node::key

key

Definition at line 34 of file minqueue_test.h.

The documentation for this struct was generated from the following file:

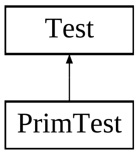
- src/queue_algorithms/min_queue/minqueue_test.h

6.39 PrimTest Class Reference

PrimTest:

```
#include <prim_test.h>
```

Inheritance diagram for PrimTest:



Public Types

- typedef Graph< PRIM_N, VertexP< int > > GType
- typedef std::function< void(VertexP< int >::VIDType v_id)> ActionType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _list_graph
- std::shared_ptr< GType > _all_edges_graph

6.39.1 Detailed Description

PrimTest:

```
PrimTest ::testing::Test TEST_F
```

Definition at line 38 of file prim_test.h.

6.39.2 Member Typedef Documentation

6.39.2.1 typedef std::function<void(VertexP<int>::VIDType v_id)> PrimTest::ActionType

Action

Definition at line 42 of file prim_test.h.

6.39.2.2 typedef Graph<PRIM_N,VertexP<int> > PrimTest::GType

```
VertexP<int>
```

Definition at line 41 of file prim_test.h.

6.39.3 Member Function Documentation

6.39.3.1 void PrimTest::SetUp ( ) [inline], [protected]

Definition at line 44 of file prim_test.h.

6.39.3.2 void PrimTest::TearDown ( ) [inline], [protected]

Definition at line 71 of file prim_test.h.

6.39.4 Member Data Documentation

6.39.4.1 std::shared_ptr<GType> PrimTest::_1e_graph [protected]

Definition at line 73 of file prim_test.h.

6.39.4.2 std::shared_ptr<GType> PrimTest::_1v_graph [protected]

Definition at line 72 of file prim_test.h.

6.39.4.3 std::shared_ptr<GType> PrimTest::_all_edges_graph [protected]

Definition at line 75 of file prim_test.h.

6.39.4.4 std::shared_ptr<GType> PrimTest::_list_graph [protected]

Definition at line 74 of file prim_test.h.

The documentation for this class was generated from the following file:

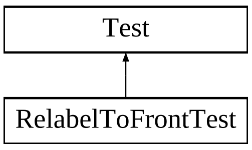
- src/graph_algorithms/minimum_spanning_tree/prim/prim_test.h

6.40 RelabelToFrontTest Class Reference

RelabelToFrontTest:

`#include <relabeltofront_test.h>`

Inheritance diagram for RelabelToFrontTest:



Public Types

- typedef Graph< RTF_N, FrontFlowVertex< int > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _graph

6.40.1 Detailed Description

RelabelToFrontTest:

`RelabelToFrontTest ::testing::Test TEST_F`

Definition at line 38 of file relabeltofront_test.h.

6.40.2 Member Typedef Documentation

6.40.2.1 typedef **Graph**<**RTF_N**,**FrontFlowVertex**<int> > **RelabelToFrontTest::GType**

`FrontFlowVertex<int>`

Definition at line 41 of file relabeltofront_test.h.

6.40.3 Member Function Documentation

6.40.3.1 void RelabelToFrontTest::SetUp ( ) `[inline], [protected]`

Definition at line 44 of file relabeltofront_test.h.

6.40.3.2 void RelabelToFrontTest::TearDown ( ) `[inline], [protected]`

Definition at line 62 of file relabeltofront_test.h.

6.40.4 Member Data Documentation

6.40.4.1 std::shared_ptr<**GType**> RelabelToFrontTest::_graph `[protected]`

26-6

Definition at line 63 of file relabeltofront_test.h.

The documentation for this class was generated from the following file:

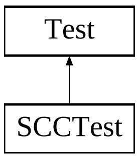
- src/graph_algorithms/max_flow/relabel_to_front/relabeltofront_test.h

6.41 SCCTest Class Reference

SCCTest:

`#include <strongconnectedcomponent_test.h>`

Inheritance diagram for SCCTest:



Public Types

- typedef Graph< SCC_N, DFS_Vertex< double > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _list_graph
- std::shared_ptr< GType > _scc_graph

6.41.1 Detailed Description

SCCTest:

`SCCTest ::testing::Test TEST_F`

Definition at line 39 of file strongconnectedcomponent_test.h.

6.41.2 Member Typedef Documentation

6.41.2.1 typedef Graph<SCC_N,DFS_Vertex<double> > SCCTest::GType

`DFS_Vertex<double>`

Definition at line 42 of file strongconnectedcomponent_test.h.

6.41.3 Member Function Documentation

6.41.3.1 void SCCTest::SetUp ( ) [inline], [protected]

Definition at line 45 of file strongconnectedcomponent_test.h.

6.41.3.2 void SCCTest::TearDown ( ) [inline], [protected]

Definition at line 65 of file strongconnectedcomponent_test.h.

6.41.4 Member Data Documentation

6.41.4.1 std::shared_ptr<GType> SCCTest::_list_graph [protected]

Definition at line 66 of file strongconnectedcomponent_test.h.

6.41.4.2 std::shared_ptr< GType > SCCTest::_scc_graph [protected]

Definition at line 67 of file strongconnectedcomponent_test.h.

The documentation for this class was generated from the following file:

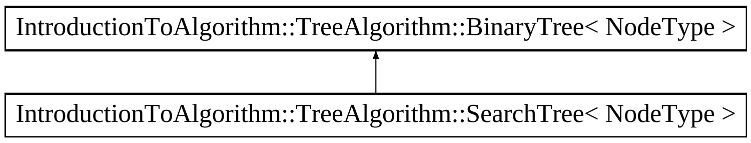
- src/graph_algorithms/basic_graph/strong_connected_component/strongconnectedcomponent_test.h

6.42 IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType > Class Template Reference

SearchTree12

`#include <searchtree.h>`

Inheritance diagram for IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >:


IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeType >
IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >

Public Types

- typedef NodeType::KeyType T

Public Member Functions

- std::shared_ptr< NodeType > search (const T &value, std::shared_ptr< NodeType > node=std::shared_↩ ptr< NodeType >())

  search:
- std::shared_ptr< NodeType > min (std::shared_ptr< NodeType > node)

  min:
- std::shared_ptr< NodeType > max (std::shared_ptr< NodeType > node)

  max:
- std::shared_ptr< NodeType > successor (std::shared_ptr< NodeType > node)

  successor:
- std::shared_ptr< NodeType > predecesor (std::shared_ptr< NodeType > node)

  predecesor:
- void insert (std::shared_ptr< NodeType > node)

  insert:
- void remove (std::shared_ptr< NodeType > node)

  remove:

Additional Inherited Members

6.42.1 Detailed Description

template<typename NodeType>class IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >

SearchTree12

Definition at line 33 of file searchtree.h.

6.42.2 Member Typedef Documentation

6.42.2.1 template<typename NodeType> typedef NodeType::KeyType IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >::T

Definition at line 36 of file searchtree.h.

6.42.3 Member Function Documentation

6.42.3.1 template<typename NodeType> void IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >::insert (std::shared_ptr< NodeType > *node*) `[inline]`

insert:

Parameters

node:	

Returns

: void

`node`

`nodenodenullptr`

O(h)O(1)h

Definition at line 191 of file searchtree.h.

6.42.3.2 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >::max (std::shared_ptr< NodeType > *node*) `[inline]`

max:

Parameters

node:	

Returns

:

O(h)O(1)h

Definition at line 99 of file searchtree.h.

6.42.3.3 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::←TreeAlgorithm::SearchTree< NodeType >::min (std::shared_ptr< NodeType > *node*) [inline]

min:

Parameters

node:	

Returns

:

O(h)O(1)h

Definition at line 75 of file searchtree.h.

6.42.3.4 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >::predecesor (std::shared_ptr< NodeType > *node*) [inline]

predecesor:

Parameters

node:	

Returns

:

`nodenode`

`node`

- `nodenode`
- `node`
  - `nodenodenode`
  - `nodenodenode->parentnodenodenode`

O(h)O(1)h

Definition at line 161 of file searchtree.h.

6.42.3.5 template<typename NodeType> void IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >::remove (std::shared_ptr< NodeType > *node*) [inline]

remove:

Parameters

node:	

Returns

: void

`nodenode`

- `node`

- `nodenode`
- `nodenode`
- `nodenodenext_node`
  - `next_nodenodenext_nodenext_nodenodenext_node next_nodenodenodenext_↩ node`
  - `next_nodenodenext_nodenodenext_nodenodenext_node`
    * `next_nodenext_node`
    * `next_nodenode`
    * `next_nodenode`

O(h)O(1)h

Definition at line 250 of file searchtree.h.

6.42.3.6 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::Tree↩ Algorithm::SearchTree< NodeType >::search (const T & *value*, std::shared_ptr< NodeType > *node* = `std::shared_ptr<`**NodeType**`>()`) `[inline]`

search:

Parameters

value	
node:	

Returns

: value

.

O(h)O(1)h

Definition at line 48 of file searchtree.h.

6.42.3.7 template<typename NodeType> std::shared_ptr<NodeType> IntroductionToAlgorithm::Tree↩ Algorithm::SearchTree< NodeType >::successor (std::shared_ptr< NodeType > *node*) `[inline]`

successor:

Parameters

node:	

Returns

:

`nodenode`

`node`

- `nodenode`
- `node`
  - `nodenodenode`

– `nodenodenode->parentnodenodenode`

O(h)O(1)h

Definition at line 127 of file searchtree.h.

The documentation for this class was generated from the following file:

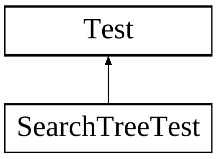
- src/tree_algorithms/searchtree/searchtree.h

6.43 SearchTreeTest Class Reference

SearchTreeTest:

`#include <searchtree_test.h>`

Inheritance diagram for SearchTreeTest:



Public Types

- typedef BinaryTreeNode< int > Node

Protected Member Functions

- SearchTreeTest ()
- void SetUp ()

  SetUp:
- void TearDown ()

  TearDown:

Protected Attributes

- SearchTree< Node > _empty_tree
- SearchTree< Node > _normal_tree

6.43.1 Detailed Description

SearchTreeTest:

`SearchTreeTest ::testing::Test TEST_F`

Definition at line 45 of file searchtree_test.h.

6.43.2 Member Typedef Documentation

6.43.2.1 typedef **BinaryTreeNode<int> SearchTreeTest::Node**

Definition at line 48 of file searchtree_test.h.

6.43.3 Constructor & Destructor Documentation

6.43.3.1 SearchTreeTest::SearchTreeTest ( ) `[inline]`, `[protected]`

Definition at line 50 of file searchtree_test.h.

6.43.4 Member Function Documentation

6.43.4.1 void SearchTreeTest::SetUp ( ) `[inline]`, `[protected]`

SetUp:

`SetUp ::testing::Test`

Definition at line 57 of file searchtree_test.h.

6.43.4.2 void SearchTreeTest::TearDown ( ) `[inline]`, `[protected]`

TearDown:

`TearDown ::testing::Test`

Definition at line 97 of file searchtree_test.h.

6.43.5 Member Data Documentation

6.43.5.1 SearchTree<Node> SearchTreeTest::_empty_tree `[protected]`

Definition at line 99 of file searchtree_test.h.

6.43.5.2 SearchTree<Node> SearchTreeTest::_normal_tree `[protected]`

Definition at line 100 of file searchtree_test.h.

The documentation for this class was generated from the following file:

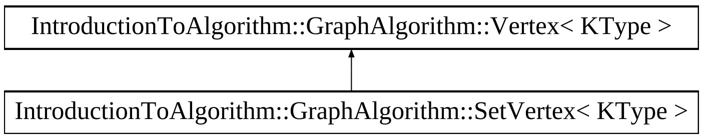
- src/tree_algorithms/searchtree/searchtree_test.h

6.44 IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType > Struct Template Reference

SetVertexnode2222.1

`#include <set_vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >

Public Types

- typedef KType KeyType
- typedef int VIDType

Public Member Functions

- SetVertex ()

- SetVertex (const KeyType &k)

  key
- SetVertex (const KeyType &k, VIDType d)

  key
- virtual std::string to_string () const

  to_string

Public Attributes

- std::shared_ptr< DisjointSetNode< SetVertex > > node

6.44.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >

SetVertexnode2222.1

VertexnodenodeDisjointSetNode<SetVertex> DisjointSetNode<SetVertex>valueSetVertex >SetVertexSet↩ VertexDisjointSetNodeDisjointSetNode valueSetVertex

Definition at line 36 of file set_vertex.h.

6.44.2 Member Typedef Documentation

6.44.2.1 template<typename KType > typedef KType IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::KeyType

Definition at line 38 of file set_vertex.h.

6.44.2.2 template<typename KType > typedef int IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::VIDType

Definition at line 39 of file set_vertex.h.

6.44.3 Constructor & Destructor Documentation

6.44.3.1 template<typename KType > IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::SetVertex ( ) [inline]

Definition at line 43 of file set_vertex.h.

6.44.3.2 template<typename KType > IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::SetVertex (const KeyType & *k*) [inline], [explicit]

key

Parameters

k:	

Definition at line 48 of file set_vertex.h.

6.44.3.3 template<typename KType > IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::SetVertex (const KeyType & *k*, VIDType *d*) [inline]

`key`

Parameters

k:	
d:	

Definition at line 54 of file set_vertex.h.

6.44.4 Member Function Documentation

6.44.4.1 template<typename KType > virtual std::string IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::to_string ( ) const [inline], [virtual]

to_string

Returns

:

`idkeyparent`

Reimplemented from IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >.

Definition at line 62 of file set_vertex.h.

6.44.5 Member Data Documentation

6.44.5.1 template<typename KType > std::shared_ptr<DisjointSetNode<SetVertex> > IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >::node

DisjointSetNode

Definition at line 75 of file set_vertex.h.

The documentation for this struct was generated from the following file:

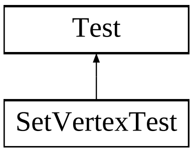
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/set_vertex.h

6.45 SetVertexTest Class Reference

SetVertexTest:

`#include <set_vertex_test.h>`

Inheritance diagram for SetVertexTest:



Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< SetVertex< double > > _default_vertex
- std::shared_ptr< SetVertex< double > > _normal_vertex

6.45.1 Detailed Description

SetVertexTest:

`SetVertexTest ::testing::Test TEST_F`

Definition at line 31 of file set_vertex_test.h.

6.45.2 Member Function Documentation

6.45.2.1 void SetVertexTest::SetUp ( ) `[inline], [protected]`

Definition at line 36 of file set_vertex_test.h.

6.45.2.2 void SetVertexTest::TearDown ( ) `[inline], [protected]`

Definition at line 41 of file set_vertex_test.h.

6.45.3 Member Data Documentation

6.45.3.1 std::shared_ptr<SetVertex<double> > SetVertexTest::_default_vertex `[protected]`

Definition at line 42 of file set_vertex_test.h.

6.45.3.2 std::shared_ptr<SetVertex<double> > SetVertexTest::_normal_vertex `[protected]`

Definition at line 43 of file set_vertex_test.h.

The documentation for this class was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/set_vertex_test.h

6.46 IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType > Class Template Reference

Sort_Heap6

```
#include <heapsort.h>
```

Public Types

- typedef std::iterator_traits< Iterator >::value_type T

Public Member Functions

- void operator() (const Iterator from, std::size_t size, CompareType compare=CompareType())

  operator()

Protected Member Functions

- void _setupHeap (CompareType compare=CompareType())

  _setupHeap:
- void _heapify (std::size_t elementIndex, CompareType compare=CompareType())

  _heapify
- std::size_t _parentIndex (std::size_t elementIndex, bool &valid)

  _parentIndex:
- std::size_t _lchildIndex (std::size_t elementIndex, bool &valid)

  _lchildIndex:
- std::size_t _rchildIndex (std::size_t elementIndex, bool &valid)

  _rchildIndex:

Private Attributes

- Iterator _from
- std::size_t _size

6.46.1 Detailed Description

template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>class IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >

Sort_Heap6

- A[p...r] 1
- O(nlogn)
-
- heapify(index)indexindexindex_heapify()
- setupHeap() heapify

Definition at line 40 of file heapsort.h.

6.46.2 Member Typedef Documentation

6.46.2.1 template<typename Iterator, typename CompareType = std::less<typename std::iterator↩ _traits<Iterator>::value_type>> typedef std::iterator_traits<Iterator>::value_type **IntroductionToAlgorithm::SortAlgorithm::Sort_Heap**< Iterator, CompareType >::T

Definition at line 43 of file heapsort.h.

6.46.3 Member Function Documentation

6.46.3.1 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_↩ type>> void **IntroductionToAlgorithm::SortAlgorithm::Sort_Heap**< Iterator, CompareType >::_heapify (std::size_t *elementIndex,* CompareType *compare =* `CompareType()`) `[inline]`, `[protected]`

_heapify

Parameters

elementIndex	:
compare	std::less<T>

Returns

void

heapify

- O(n)
-

Definition at line 102 of file heapsort.h.

6.46.3.2 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value↩ _type>> std::size_t **IntroductionToAlgorithm::SortAlgorithm::Sort_Heap**< Iterator, CompareType >::_lchildIndex (std::size_t *elementIndex,* bool & *valid*) `[inline]`, `[protected]`

_lchildIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex/2)+1

- 01
-

Definition at line 162 of file heapsort.h.

6.46.3.3 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> std::size_t IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::_parentIndex (std::size_t *elementIndex,* bool & *valid*) [inline], [protected]

_parentIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex-1)/2

-

Definition at line 139 of file heapsort.h.

6.46.3.4 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> std::size_t IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::_rchildIndex (std::size_t *elementIndex*, bool & *valid*) [inline], [protected]

_rchildIndex:

Parameters

elementIndex	:
valid	bool&

Returns

(std::size_t)

elementIndex(elementIndex/2)+2

- 012
-

Definition at line 190 of file heapsort.h.

6.46.3.5 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> void IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::_setupHeap (CompareType *compare* = `CompareType()`) [inline], [protected]

_setupHeap:

Parameters

compare	std::less<T>

Returns

void

heapify

- O(nlogn)
-

Definition at line 79 of file heapsort.h.

6.46.3.6 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_↩ type>> void IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::operator() (const Iterator *from*, std::size_t *size*, CompareType *compare* = `CompareType()`) `[inline]`

operator()

Parameters

from	:
size	
compare	std::less<T>

Returns

void

_setupHeap()

- O(nlogn)
-

Definition at line 56 of file heapsort.h.

6.46.4 Member Data Documentation

6.46.4.1 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_←type>> Iterator IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::_from [private]

Definition at line 206 of file heapsort.h.

6.46.4.2 template<typename Iterator, typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_←type>> std::size_t IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >::_size [private]

Definition at line 207 of file heapsort.h.

The documentation for this class was generated from the following file:

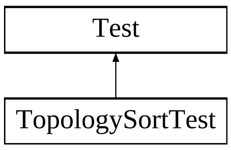
- src/sort_algorithms/heap_sort/heapsort.h

6.47 TopologySortTest Class Reference

TopologySortTest:

`#include <topologysort_test.h>`

Inheritance diagram for TopologySortTest:



Public Types

- typedef Graph< TPS_N, DFS_Vertex< double > > GType

Protected Member Functions

- void SetUp ()
- void TearDown ()

Protected Attributes

- std::shared_ptr< GType > _1v_graph
- std::shared_ptr< GType > _1e_graph
- std::shared_ptr< GType > _list_graph

6.47.1 Detailed Description

TopologySortTest:

`TopologySortTest ::testing::Test TEST_F`

Definition at line 40 of file topologysort_test.h.

6.47.2 Member Typedef Documentation

6.47.2.1 typedef Graph<TPS_N,DFS_Vertex<double> > TopologySortTest::GType

`DFS_Vertex<double>`

Definition at line 43 of file topologysort_test.h.

6.47.3 Member Function Documentation

6.47.3.1 void TopologySortTest::SetUp ( ) [inline], [protected]

Definition at line 46 of file topologysort_test.h.

6.47.3.2 void TopologySortTest::TearDown ( ) [inline], [protected]

Definition at line 64 of file topologysort_test.h.

6.47.4 Member Data Documentation

6.47.4.1 std::shared_ptr<GType> TopologySortTest::_1e_graph [protected]

Definition at line 66 of file topologysort_test.h.

6.47.4.2 std::shared_ptr<GType> TopologySortTest::_1v_graph [protected]

Definition at line 65 of file topologysort_test.h.

6.47.4.3 std::shared_ptr<GType> TopologySortTest::_list_graph [protected]

Definition at line 67 of file topologysort_test.h.

The documentation for this class was generated from the following file:

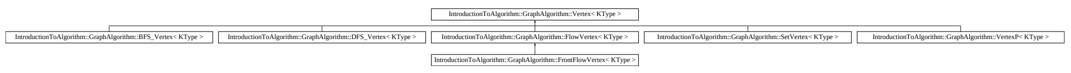
- src/graph_algorithms/basic_graph/topology_sort/topologysort_test.h

6.48 IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType > Struct Template Reference

Vertex2222.1

`#include <vertex.h>`

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >
IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >

Public Types

- typedef KType KeyType
- typedef int VIDType

Public Member Functions

- Vertex ()

  `keyKType()`-1
- Vertex (const KeyType &k)

  `key`
- Vertex (const KeyType &k, VIDType d)

  `key`
- virtual std::string to_string () const

  to_string

Public Attributes

- KeyType key
- const VIDType id

6.48.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >

Vertex2222.1

- `key`:
- `id`:0`const int`

`id`-1,`keyT()`

Definition at line 38 of file vertex.h.

6.48.2 Member Typedef Documentation

6.48.2.1 template<typename KType> typedef KType IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::KeyType

Definition at line 40 of file vertex.h.

6.48.2.2 template<typename KType> typedef int IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::VIDType

Definition at line 41 of file vertex.h.

6.48.3 Constructor & Destructor Documentation

6.48.3.1 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::Vertex ( ) `[inline]`

`keyKType()`-1

Definition at line 44 of file vertex.h.

6.48.3.2 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::Vertex (const KeyType & *k*) `[inline]`, `[explicit]`

`key`

Parameters

k:	

Definition at line 49 of file vertex.h.

6.48.3.3 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::Vertex (const KeyType & *k*, VIDType *d*) `[inline]`

`key`

Parameters

k:	
d:	

Definition at line 55 of file vertex.h.

6.48.4 Member Function Documentation

6.48.4.1 template<typename KType> virtual std::string IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::to_string ( ) const `[inline]`, `[virtual]`

to_string

Returns

:

```
idkey
```

Reimplemented in IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >, IntroductionTo↩ Algorithm::GraphAlgorithm::SetVertex< KType >, IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >, and IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >.

Definition at line 63 of file vertex.h.

6.48.5 Member Data Documentation

6.48.5.1 template<typename KType> const VIDType IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::id

idid

Definition at line 70 of file vertex.h.

6.48.5.2 template<typename KType> KeyType IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >::key

Definition at line 69 of file vertex.h.

The documentation for this struct was generated from the following file:

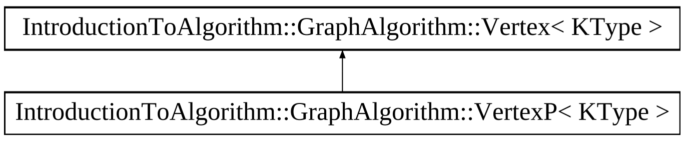
- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertex.h

6.49 IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType > Struct Template Reference

VertexPparent2222.1

```
#include <vertexp.h>
```

Inheritance diagram for IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >:


IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >
IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >

Public Types

- typedef KType KeyType
- typedef int VIDType

Public Member Functions

- VertexP ()

- VertexP (const KeyType &k)

key
- VertexP (const KeyType &k, VIDType d)

  key
- virtual std::string to_string () const

  to_string

Public Attributes

- std::shared_ptr< VertexP > parent

6.49.1 Detailed Description

template<typename KType>struct IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >

VertexPparent2222.1

Vertexparent

Definition at line 31 of file vertexp.h.

6.49.2 Member Typedef Documentation

6.49.2.1 template<typename KType> typedef KType IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::KeyType

Definition at line 33 of file vertexp.h.

6.49.2.2 template<typename KType> typedef int IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::VIDType

Definition at line 34 of file vertexp.h.

6.49.3 Constructor & Destructor Documentation

6.49.3.1 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::VertexP ( ) `[inline]`

Definition at line 37 of file vertexp.h.

6.49.3.2 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::VertexP (const KeyType & *k*) `[inline]`, `[explicit]`

`key`

Parameters

k:	

Definition at line 42 of file vertexp.h.

6.49.3.3 template<typename KType> IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::VertexP (const KeyType & *k*, VIDType *d*) [inline]

`key`

Parameters

k:	
d:	

Definition at line 48 of file vertexp.h.

6.49.4 Member Function Documentation

6.49.4.1 template<typename KType> virtual std::string IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >::to_string ( ) const [inline], [virtual]

to_string

Returns

:

```
idkeyparent
```

Reimplemented from IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >.

Definition at line 56 of file vertexp.h.

6.49.5 Member Data Documentation

6.49.5.1 template<typename KType> std::shared_ptr<VertexP> IntroductionToAlgorithm::GraphAlgorithm::↩ VertexP< KType >::parent

Definition at line 64 of file vertexp.h.

The documentation for this struct was generated from the following file:

- src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertexp.h

Chapter 7

File Documentation

7.1 src/dynamic_programming_algorithms/lcs/longest_common_subsequence.h File Reference

```
#include <type_traits>
#include <vector>
#include <iostream>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::DynamicProgrammingAlgorithm

  Namespace of DynamicProgrammingAlgorithm.

Functions

- template< typename Iterator , typename OutIterator >
  std::size_t IntroductionToAlgorithm::DynamicProgrammingAlgorithm::make_LCS (const Iterator begin, const Iterator end, const std::vector< std::vector< int >> &flag_matrix, typename std::iterator_traits< Iterator >↩ ::difference_type seq1_index, typename std::iterator_traits< Iterator >::difference_type seq2_index, Out↩ Iterator &out_begin)

  make_LCS
- template< typename Iterator1 , typename Iterator2 , typename OutIterator >
  std::size_t IntroductionToAlgorithm::DynamicProgrammingAlgorithm::longest_common_subsequence (const Iterator1 first_begin, const Iterator1 first_end, const Iterator2 second_begin, const Iterator2 second_end, OutIterator out_begin)

  longest_common_subsequence 159.4

7.2 src/dynamic_programming_algorithms/lcs/longest_common_subsequence_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "longest_common_subsequence.h"
```

Functions

- TEST (test_longest_common_subsequence, test1)

    longest_common_subsequence_test

7.2.1 Function Documentation

7.2.1.1 TEST (test_longest_common_subsequence , test1)

longest_common_subsequence_test

s1s2s1s2s1s2

Definition at line 30 of file longest_common_subsequence_test.h.

7.3 src/graph_algorithms/all_node_pair_shortest_path/floyd_warshall/floyd_warshall.h File Reference

```
#include <memory>
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

    Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  std::pair< std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::↩NUM >, std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > > IntroductionToAlgorithm::GraphAlgorithm::floyd_warshall (std::shared_ptr< GraphType > graph)

    floyd_warshallfloyd_warshall2525.2

7.4 src/graph_algorithms/all_node_pair_shortest_path/floyd_warshall/floyd_warshall_↩test.h File Reference

```
#include "src/google_test/gtest.h"
#include "floyd_warshall.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertex.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
```

Classes

- class FloydWarshallTest

    FloydWarshallTest:

Namespaces

- anonymous_namespace{floyd_warshall_test.h}

Functions

- TEST_F (FloydWarshallTest, test_floyd_warshall)

  FloydWarshallTest: floyd_warshall

Variables

- const int anonymous_namespace{floyd_warshall_test.h}::FW_N = 5

7.4.1 Function Documentation

7.4.1.1 TEST_F (FloydWarshallTest , test_floyd_warshall)

FloydWarshallTest: floyd_warshall

`test_floyd_warshall` floyd_warshall

Definition at line 73 of file floyd_warshall_test.h.

7.5 src/graph_algorithms/all_node_pair_shortest_path/johnson/johnson.h File Reference

```
#include <memory>
#include "src/header.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "../../single_source_shortest_path/bellman_ford/bellmanford.h"
#include "../../single_source_shortest_path/dijkstra/dijkstra.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template< typename GraphType >
  std::shared_ptr< Graph< GraphType::NUM+1, typename GraphType::VertexType > > IntroductionTo↩ Algorithm::GraphAlgorithm::graph_plus_1v (std::shared_ptr< GraphType > graph)

  graph_plus_1vgraph2525.2
- template< typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::johnson (std::shared_ptr< GraphType > graph)

  johnsonjohnson2525.3

7.6 src/graph_algorithms/all_node_pair_shortest_path/johnson/johnson_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "johnson.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertexp.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
```

Classes

- class JohnsonTest

  JohnsonTest:

Namespaces

- anonymous_namespace{johnson_test.h}

Functions

- TEST_F (JohnsonTest, test_johnson)

  JohnsonTest: johnson

Variables

- const int anonymous_namespace{johnson_test.h}::JS_N = 5

7.6.1 Function Documentation

7.6.1.1 TEST_F (JohnsonTest , test_johnson)

JohnsonTest: johnson

`test_johnson` johnson

Definition at line 73 of file johnson_test.h.

7.7 src/graph_algorithms/all_node_pair_shortest_path/matrix_shortest_path/matrix_↩ shortest_path.h File Reference

```
#include <memory>
#include <array>
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename MatrixType >
  MatrixType IntroductionToAlgorithm::GraphAlgorithm::extend_path (const MatrixType &L, const MatrixType &W)

  extend_path2525.1
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::matrix_shortest_path (std::shared_ptr< GraphType > graph)

  matrix_shortest_path2525.1
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType,GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::matrix_shortest_path_fast (std::shared_ptr< GraphType > graph)

  matrix_shortest_path2525.1

7.8 src/graph_algorithms/all_node_pair_shortest_path/matrix_shortest_path/matrix_↵ shortest_path_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "matrix_shortest_path.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertex.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
```

Classes

- class MatrixShortestPathTest

  MatrixShortestPathTest:

Namespaces

- anonymous_namespace{matrix_shortest_path_test.h}

Functions

- TEST_F (MatrixShortestPathTest, test_matrix_shortest_path)

  MatrixShortestPathTest: matrix_shortest_path
- TEST_F (MatrixShortestPathTest, test_matrix_shortest_path_fast)

  MatrixShortestPathTest: matrix_shortest_path_fast

Variables

- const int anonymous_namespace{matrix_shortest_path_test.h}::MT_N = 5

7.8.1 Function Documentation

7.8.1.1 TEST_F (MatrixShortestPathTest , test_matrix_shortest_path)

MatrixShortestPathTest: matrix_shortest_path

`test_matrix_shortest_path` matrix_shortest_path

Definition at line 75 of file matrix_shortest_path_test.h.

7.8.1.2 TEST_F (MatrixShortestPathTest , test_matrix_shortest_path_fast)

MatrixShortestPathTest: matrix_shortest_path_fast

`test_matrix_shortest_path_fast` matrix_shortest_path_fast

Definition at line 91 of file matrix_shortest_path_test.h.

7.9 src/graph_algorithms/basic_graph/connected_component/connectedcomponent.h File Reference

```
#include "src/set_algorithms/disjoint_set/disjointset.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::connected_component (std::shared_ptr< GraphType > graph)

  connected_component2121.1
- template<typename GraphType >
  bool IntroductionToAlgorithm::GraphAlgorithm::same_component (std::shared_ptr< GraphType > graph, typename GraphType::VIDType id1, typename GraphType::VIDType id2)

  same_component2121.1

7.10 src/graph_algorithms/basic_graph/connected_component/connectedcomponent_↩ test.h File Reference

```
#include "src/google_test/gtest.h"
#include "connectedcomponent.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "../../basic_graph/graph_representation/graph_vertex/set_vertex.h"
```

Classes

- class ConnectedComponentTest

  ConnectedComponentTest:

Namespaces

- anonymous_namespace{connectedcomponent_test.h}

Functions

- TEST_F (ConnectedComponentTest, test_connected_component)

  test_connected_componentconnected_component
- TEST_F (ConnectedComponentTest, test_same_component)

  test_same_componentsame_component

Variables

- const int anonymous_namespace{connectedcomponent_test.h}::C_NUM =10

7.10.1 Function Documentation

7.10.1.1 TEST_F(ConnectedComponentTest , test_connected_component)

test_connected_componentconnected_component

```
test_connected_componentconnected_component()
```

Definition at line 68 of file connectedcomponent_test.h.

7.10.1.2 TEST_F(ConnectedComponentTest , test_same_component)

test_same_componentsame_component

```
test_same_componentsame_component()
```

Definition at line 83 of file connectedcomponent_test.h.

7.11 src/graph_algorithms/basic_graph/graph_bfs/bfs.h File Reference

```
#include <memory>
#include <queue>
#include <functional>
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::breadth_first_search (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id, std::function< void(typename GraphType::VIDType)> pre_↩ action=[ ](typename GraphType::VIDType){}, std::function< void(typename GraphType::VIDType)> post_↩ action=[ ](typename GraphType::VIDType){})

  breadth_first_search2222.2

7.12 src/graph_algorithms/basic_graph/graph_bfs/bfs_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "bfs.h"
#include "../graph_representation/graph/graph.h"
#include "../graph_representation/graph_vertex/bfs_vertex.h"
```

Classes

- class BFSTest

  BFSTest:

Namespaces

- anonymous_namespace{bfs_test.h}

Functions

- TEST_F (BFSTest, test_bfs)

  test_bfs:breadth_first_search
- TEST_F (BFSTest, test_get_path)

  test_get_path:get_path

Variables

- const int anonymous_namespace{bfs_test.h}::BFS_N = 10

7.12.1 Function Documentation

7.12.1.1 TEST_F (**BFSTest** , test_bfs)

test_bfs:breadth_first_search

```
breadth_first_search
```

Definition at line 74 of file bfs_test.h.

7.12.1.2 TEST_F (**BFSTest** , test_get_path)

test_get_path:get_path

```
get_path
```

Definition at line 119 of file bfs_test.h.

7.13 src/graph_algorithms/basic_graph/graph_dfs/dfs.h File Reference

```
#include <memory>
#include <functional>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template< typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::visit (std::shared_ptr< GraphType > graph, typename GraphType::VIDType v_id, int &time, std::function< void(typename GraphType::VIDType, int)> pre_↩ action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> post_action=[ ](typename GraphType::VIDType, int){})

  visit2222.3
- template< typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::depth_first_search (std::shared_ptr< GraphType > graph, std::function< void(typename GraphType::VIDType, int)> pre_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> post_action=[ ](typename GraphType::↩ VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> pre_root_action=[ ](typename GraphType::VIDType, int){}, std::function< void(typename GraphType::VIDType, int)> post_root_↩ action=[ ](typename GraphType::VIDType, int){}, const std::vector< typename GraphType::VIDType > &search_order=std::vector< typename GraphType::VIDType >())

  depth_first_search2222.3

7.14 src/graph_algorithms/basic_graph/graph_dfs/dfs_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "dfs.h"
#include "src/header.h"
#include "../graph_representation/graph/graph.h"
#include "../graph_representation/graph_vertex/dfs_vertex.h"
```

Classes

- class DFSTest

  DFSTest:

Namespaces

- anonymous_namespace{dfs_test.h}

Functions

- TEST_F (DFSTest, test_dfs)

  test_dfs:depth_first_search
- TEST_F (DFSTest, test_get_path)

  test_get_path:get_path

Variables

- const int anonymous_namespace{dfs_test.h}::DFS_N = 10

7.14.1 Function Documentation

7.14.1.1 TEST_F (DFSTest , test_dfs)

test_dfs:depth_first_search

```
depth_first_search
```

Definition at line 84 of file dfs_test.h.

7.14.1.2 TEST_F (DFSTest , test_get_path)

test_get_path:get_path

```
get_path
```

Definition at line 155 of file dfs_test.h.

7.15 src/graph_algorithms/basic_graph/graph_representation/adjlist_graph/adjlistgraph.h File Reference

```
#include <vector>
#include <array>
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::ADJListGraph< N >

  ADJListGraph2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.16 src/graph_algorithms/basic_graph/graph_representation/adjlist_graph/adjlistgraph↩_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "adjlistgraph.h"
#include <memory>
```

Classes

- class GraphADJListTest

    GraphADJListTest:

Namespaces

- anonymous_namespace{adjlistgraph_test.h}

Functions

- TEST_F (GraphADJListTest, test_weight)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_has_edge)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_add_edge)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_add_edges)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_adjust_edge)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_edge_tuples)

    adjlist_graph_test:ADJListGraph
- TEST_F (GraphADJListTest, test_vertex_edge_tuples)

    adjlist_graph_test:ADJListGraph

Variables

- const int anonymous_namespace{adjlistgraph_test.h}::ADJ_NUM =10

7.16.1 Function Documentation

7.16.1.1 TEST_F (**GraphADJListTest** , test_weight)

adjlist_graph_test:ADJListGraph

`weight`

Definition at line 51 of file adjlistgraph_test.h.

7.16.1.2 TEST_F (**GraphADJListTest** , test_has_edge)

adjlist_graph_test:ADJListGraph

`has_edge`

Definition at line 63 of file adjlistgraph_test.h.

7.16.1.3 TEST_F (**GraphADJListTest** , test_add_edge)

adjlist_graph_test:ADJListGraph

`add_edge`

Definition at line 78 of file adjlistgraph_test.h.

7.16.1.4 TEST_F (GraphADJListTest , test_add_edges)

adjlist_graph_test:ADJListGraph

`add_edges`

Definition at line 98 of file adjlistgraph_test.h.

7.16.1.5 TEST_F (GraphADJListTest , test_adjust_edge)

adjlist_graph_test:ADJListGraph

`adjust_edge`

Definition at line 117 of file adjlistgraph_test.h.

7.16.1.6 TEST_F (GraphADJListTest , test_edge_tuples)

adjlist_graph_test:ADJListGraph

`edge_tuples`

Definition at line 139 of file adjlistgraph_test.h.

7.16.1.7 TEST_F (GraphADJListTest , test_vertex_edge_tuples)

adjlist_graph_test:ADJListGraph

`vertex_edge_tuples`

Definition at line 156 of file adjlistgraph_test.h.

7.17 src/graph_algorithms/basic_graph/graph_representation/graph/graph.h File Reference

```
#include "../matrix_graph/matrixgraph.h"
#include "../adjlist_graph/adjlistgraph.h"
#include <array>
#include <memory>
#include <assert.h>
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::Graph< N, VType >

  Graph2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.18 src/graph_algorithms/basic_graph/graph_representation/graph/graph_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "../graph_vertex/vertex.h"
#include "graph.h"
```

Classes

- class GraphTest

  GraphTest:

Namespaces

- anonymous_namespace{graph_test.h}

Functions

- TEST_F (GraphTest, test_add_vertex)

  graph_test:Graph
- TEST_F (GraphTest, test_add_vertex_with_id)

  graph_test:Graph
- TEST_F (GraphTest, test_modify_vertex)

  graph_test:Graph
- TEST_F (GraphTest, test_weight)

  graph_test:Graph
- TEST_F (GraphTest, test_has_edge)

  graph_test:Graph
- TEST_F (GraphTest, test_add_edge)

  graph_test:Graph
- TEST_F (GraphTest, test_add_edges)

  graph_test:Graph
- TEST_F (GraphTest, test_adjust_edge)

  graph_test:Graph
- TEST_F (GraphTest, test_edge_tuples)

  graph_test:Graph
- TEST_F (GraphTest, test_vertex_edge_tuples)

  graph_test:Graph
- TEST_F (GraphTest, test_inverse)

  graph_test:Graph

Variables

- const int anonymous_namespace{graph_test.h}::G_N = 10

7.18.1 Function Documentation

7.18.1.1 TEST_F (GraphTest , test_add_vertex)

graph_test:Graph

`add_vertex`

Definition at line 70 of file graph_test.h.

7.18.1.2 TEST_F (GraphTest , test_add_vertex_with_id)

graph_test:Graph

`add_vertexadd_vertexid`

Definition at line 90 of file graph_test.h.

7.18.1.3 TEST_F (GraphTest , test_modify_vertex)

graph_test:Graph

`modify_vertex`

Definition at line 113 of file graph_test.h.

7.18.1.4 TEST_F (GraphTest , test_weight)

graph_test:Graph

`weight`

Definition at line 135 of file graph_test.h.

7.18.1.5 TEST_F (GraphTest , test_has_edge)

graph_test:Graph

`has_edge`

Definition at line 153 of file graph_test.h.

7.18.1.6 TEST_F (GraphTest , test_add_edge)

graph_test:Graph

`add_edge`

Definition at line 171 of file graph_test.h.

7.18.1.7 TEST_F (GraphTest , test_add_edges)

graph_test:Graph

`add_edges`

Definition at line 194 of file graph_test.h.

7.18.1.8 TEST_F (GraphTest , test_adjust_edge)

graph_test:Graph

`adjust_edge`

Definition at line 212 of file graph_test.h.

7.18.1.9 TEST_F (GraphTest , test_edge_tuples)

graph_test:Graph

`edge_tuples`

Definition at line 234 of file graph_test.h.

7.18.1.10 TEST_F (GraphTest , test_vertex_edge_tuples)

graph_test:Graph

`vertex_edge_tuples`

Definition at line 250 of file graph_test.h.

7.18.1.11 TEST_F (GraphTest , test_inverse)

graph_test:Graph

`inverse`

Definition at line 271 of file graph_test.h.

7.19 src/graph_algorithms/basic_graph/graph_representation/graph_edge/edge.h File Reference

```
#include <tuple>
#include <memory>
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::Edge< VType >

  Edge2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.20 src/graph_algorithms/basic_graph/graph_representation/graph_edge/edge_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "edge.h"
#include "../graph_vertex/vertex.h"
```

Classes

- class EdgeTest

  EdgeTest:`Edge`

Functions

- TEST_F (EdgeTest, test_data_member)

  test_edge
- TEST_F (EdgeTest, test_to_string)

  test_edge
- TEST_F (EdgeTest, test_edge_tuple)

  test_edge

7.20.1 Function Documentation

7.20.1.1 TEST_F (**EdgeTest** , test_data_member)

test_edge

Definition at line 49 of file edge_test.h.

7.20.1.2 TEST_F (**EdgeTest** , test_to_string)

test_edge

`to_string`

Definition at line 58 of file edge_test.h.

7.20.1.3 TEST_F (**EdgeTest** , test_edge_tuple)

test_edge

`edge_tuple`

Definition at line 66 of file edge_test.h.

7.21 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/bfs_vertex.h File Reference

```
#include "vertex.h"
#include "src/header.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::BFS_Vertex< KType >

  BFS_Vertex2222.2

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.22 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/bfs_vertex↩ _test.h File Reference

```
#include "src/google_test/gtest.h"
#include "bfs_vertex.h"
```

Classes

- class BFSVertexTest

  BFSVertexTest:

Functions

- TEST_F (BFSVertexTest, test_data_member)

  bfs_vertex_test:BFSVertex
- TEST_F (BFSVertexTest, test_set_source)

  bfs_vertex_test:BFSVertex
- TEST_F (BFSVertexTest, test_set_found)

  bfs_vertex_test:BFSVertex
- TEST_F (BFSVertexTest, test_to_string)

  bfs_vertex_test:BFSVertex

7.22.1 Function Documentation

7.22.1.1 TEST_F (BFSVertexTest , test_data_member)

bfs_vertex_test:BFSVertex

`BFSVertex`

Definition at line 50 of file bfs_vertex_test.h.

7.22.1.2 TEST_F (BFSVertexTest , test_set_source)

bfs_vertex_test:BFSVertex

`BFSVertexset_source`

Definition at line 68 of file bfs_vertex_test.h.

7.22.1.3 TEST_F (BFSVertexTest , test_set_found)

bfs_vertex_test:BFSVertex

```
BFSVertexset_found
```

Definition at line 83 of file bfs_vertex_test.h.

7.22.1.4 TEST_F (BFSVertexTest , test_to_string)

bfs_vertex_test:BFSVertex

```
BFSVertexto_string
```

Definition at line 98 of file bfs_vertex_test.h.

7.23 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/dfs_vertex.h File Reference

```
#include "vertex.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::DFS_Vertex< KType >

    DFS_Vertex2222.3

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

    Namespace of GraphAlgorithm.

7.24 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/dfs_vertex←_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "dfs_vertex.h"
```

Classes

- class DFSVertexTest

    DFSVertexTest:

Functions

- TEST_F (DFSVertexTest, test_data_member)

    dfs_vertex_test:DFSVertex

- TEST_F (DFSVertexTest, test_set_disovered)

  dfs_vertex_test:DFSVertex
- TEST_F (DFSVertexTest, test_set_finished)

  dfs_vertex_test:DFSVertex
- TEST_F (DFSVertexTest, test_to_string)

  dfs_vertex_test:DFSVertex

7.24.1 Function Documentation

7.24.1.1 TEST_F (DFSVertexTest , test_data_member)

dfs_vertex_test:DFSVertex

```
DFSVertex
```

Definition at line 50 of file dfs_vertex_test.h.

7.24.1.2 TEST_F (DFSVertexTest , test_set_disovered)

dfs_vertex_test:DFSVertex

```
DFSVertexset_disovered
```

Definition at line 72 of file dfs_vertex_test.h.

7.24.1.3 TEST_F (DFSVertexTest , test_set_finished)

dfs_vertex_test:DFSVertex

```
DFSVertexset_finished
```

Definition at line 85 of file dfs_vertex_test.h.

7.24.1.4 TEST_F (DFSVertexTest , test_to_string)

dfs_vertex_test:DFSVertex

```
DFSVertexto_string
```

Definition at line 98 of file dfs_vertex_test.h.

7.25 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/flow_↩ vertex.h File Reference

```
#include "vertex.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::FlowVertex< KType >

  FlowVertex-2626.4

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.26 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/flow_↩ vertex_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "flow_vertex.h"
```

Functions

- TEST (test_flowvertex, flowvertex_test)

  flowvertex_test FlowVertex

7.26.1 Function Documentation

7.26.1.1 TEST (test_flowvertex , flowvertex_test)

flowvertex_test FlowVertex

FlowVertex

Definition at line 30 of file flow_vertex_test.h.

7.27 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow↩ _vertex.h File Reference

```
#include "flow_vertex.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::List< NodeType >

  List
- struct IntroductionToAlgorithm::GraphAlgorithm::ListNode< ValueType >

  ListNode
- struct IntroductionToAlgorithm::GraphAlgorithm::FrontFlowVertex< KType >

  FrontFlowVertexrelabel_to_front2626.4

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.28 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/front_flow↩_vertex_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "front_flow_vertex.h"
```

Classes

- class FrontFlowVertexTest

Namespaces

- anonymous_namespace{front_flow_vertex_test.h}

Functions

- TEST_F (FrontFlowVertexTest, test_FrontFlowVertex_member)

  FrontFlowVertexTest FrontFlowVertex.
- TEST_F (FrontFlowVertexTest, test_node)

  FrontFlowVertexTest FrontFlowVertex.
- TEST_F (FrontFlowVertexTest, test_list)

  FrontFlowVertexTest FrontFlowVertex.

Variables

- const int anonymous_namespace{front_flow_vertex_test.h}::FFV_NUM =5

7.28.1 Function Documentation

7.28.1.1 TEST_F (**FrontFlowVertexTest** , test_FrontFlowVertex_member)

FrontFlowVertexTest FrontFlowVertex.

FrontFlowVertex

Definition at line 63 of file front_flow_vertex_test.h.

7.28.1.2 TEST_F (**FrontFlowVertexTest** , test_node)

FrontFlowVertexTest FrontFlowVertex.

ListNode

Definition at line 82 of file front_flow_vertex_test.h.

7.28.1.3 TEST_F (**FrontFlowVertexTest** , test_list)

FrontFlowVertexTest FrontFlowVertex.

List

Definition at line 96 of file front_flow_vertex_test.h.

7.29 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/set_vertex.h File Reference

```
#include "vertex.h"
#include "src/set_algorithms/disjoint_set/disjointset.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::SetVertex< KType >

  SetVertexnode2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.30 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/set_vertex←_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "set_vertex.h"
```

Classes

- class SetVertexTest

  SetVertexTest:

Functions

- TEST_F (SetVertexTest, test_data_member)

  test_data_memberSetVertex
- TEST_F (SetVertexTest, test_to_string)

  test_to_stringSetVertex

7.30.1 Function Documentation

7.30.1.1 TEST_F (SetVertexTest , test_data_member)

test_data_memberSetVertex

`test_data_member`SetVertex

Definition at line 50 of file set_vertex_test.h.

7.30.1.2 TEST_F (SetVertexTest , test_to_string)

test_to_stringSetVertex

`test_to_string`SetVertex`to_string()`

Definition at line 65 of file set_vertex_test.h.

7.31 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertex.h File Reference

```
#include <memory>
#include <sstream>
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::Vertex< KType >

  Vertex2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.32 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertex_↩ test.h File Reference

```
#include "src/google_test/gtest.h"
#include "vertex.h"
#include "src/header.h"
```

Functions

- TEST (test_unlimit, unlimit_is_unlimit_test)

  test_unlimit unlimitis_unlimit
- TEST (test_vertex, vertex_test)

  test_vertexVertex

7.32.1 Function Documentation

7.32.1.1 TEST (test_unlimit , unlimit_is_unlimit_test)

test_unlimit unlimitis_unlimit

Definition at line 32 of file vertex_test.h.

7.32.1.2 TEST (test_vertex , vertex_test)

test_vertexVertex

Vertex

Definition at line 51 of file vertex_test.h.

7.33 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertexp.h File Reference

```
#include "vertex.h"
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::VertexP< KType >

  VertexPparent2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.34 src/graph_algorithms/basic_graph/graph_representation/graph_vertex/vertexp_↩ test.h File Reference

```
#include "src/google_test/gtest.h"
#include "vertexp.h"
```

Functions

- TEST (test_vertex_p, vertexp_test)

  test_vertex_pVertexP

7.34.1 Function Documentation

7.34.1.1 TEST (test_vertex_p , vertexp_test)

test_vertex_pVertexP

VertexP

Definition at line 29 of file vertexp_test.h.

7.35 src/graph_algorithms/basic_graph/graph_representation/matrix_graph/matrixgraph.h File Reference

```
#include <array>
#include <map>
```

Classes

- struct IntroductionToAlgorithm::GraphAlgorithm::MatrixGraph< N >

  MatrixGraph2222.1

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

7.36 src/graph_algorithms/basic_graph/graph_representation/matrix_graph/matrixgraph↩_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "matrixgraph.h"
#include <memory>
```

Classes

- class GraphMatrixTest

  GraphMatrixTest:

Namespaces

- anonymous_namespace{matrixgraph_test.h}

Functions

- TEST_F (GraphMatrixTest, test_weight)

  matrix_graph_test:MatrixGraph
- TEST_F (GraphMatrixTest, test_has_edge)

  matrix_graph_test:MatrixGraph
- TEST_F (GraphMatrixTest, test_add_edge)

  matrix_graph_test:MatrixGraph
- TEST_F (GraphMatrixTest, test_add_edges)

  matrix_graph_test:MatrixGraph
- TEST_F (GraphMatrixTest, test_adjust_edge)

  matrix_graph_test:MatrixGraph
- TEST_F (GraphMatrixTest, test_edge_tuples)

matrix_graph_test:MatrixGraph

- TEST_F (GraphMatrixTest, test_vertex_edge_tuples)

  matrix_graph_test:MatrixGraph

Variables

- const int anonymous_namespace{matrixgraph_test.h}::MTXNUM =10

7.36.1 Function Documentation

7.36.1.1 TEST_F (**GraphMatrixTest** , test_weight)

matrix_graph_test:MatrixGraph

`weight`

Definition at line 50 of file matrixgraph_test.h.

7.36.1.2 TEST_F (**GraphMatrixTest** , test_has_edge)

matrix_graph_test:MatrixGraph

`has_edge`

Definition at line 61 of file matrixgraph_test.h.

7.36.1.3 TEST_F (**GraphMatrixTest** , test_add_edge)

matrix_graph_test:MatrixGraph

`add_edge`

Definition at line 76 of file matrixgraph_test.h.

7.36.1.4 TEST_F (**GraphMatrixTest** , test_add_edges)

matrix_graph_test:MatrixGraph

`add_edges`

Definition at line 95 of file matrixgraph_test.h.

7.36.1.5 TEST_F (**GraphMatrixTest** , test_adjust_edge)

matrix_graph_test:MatrixGraph

`adjust_edge`

Definition at line 114 of file matrixgraph_test.h.

7.36.1.6 TEST_F (**GraphMatrixTest** , test_edge_tuples)

matrix_graph_test:MatrixGraph

`edge_tuples`

Definition at line 136 of file matrixgraph_test.h.

7.36.1.7 TEST_F (GraphMatrixTest , test_vertex_edge_tuples)

matrix_graph_test:MatrixGraph

`vertex_edge_tuples`

Definition at line 154 of file matrixgraph_test.h.

7.37 src/graph_algorithms/basic_graph/strong_connected_component/strongconnectedcomponent.h File Reference

```
#include "../graph_dfs/dfs.h"
#include <set>
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

    Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  const std::vector< std::vector< typename GraphType::VIDType > > IntroductionToAlgorithm::Graph↩Algorithm::scc (std::shared_ptr< GraphType > graph)

    scc2222.5

7.38 src/graph_algorithms/basic_graph/strong_connected_component/strongconnectedcomponent↩_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "strongconnectedcomponent.h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph/graph.↩
h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph_↩
vertex/dfs_vertex.h"
```

Classes

- class SCCTest

    SCCTest:

Namespaces

- anonymous_namespace{strongconnectedcomponent_test.h}

Functions

- TEST_F (SCCTest, test_scc)

  test_scc:scc

Variables

- const int anonymous_namespace{strongconnectedcomponent_test.h}::SCC_N = 10

7.38.1 Function Documentation

7.38.1.1 TEST_F (SCCTest , test_scc)

test_scc:scc

`scc`

Definition at line 76 of file strongconnectedcomponent_test.h.

7.39 src/graph_algorithms/basic_graph/topology_sort/topologysort.h File Reference

```
#include "../graph_dfs/dfs.h"
#include <vector>
#include <functional>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  std::vector< typename GraphType::VIDType > IntroductionToAlgorithm::GraphAlgorithm::topology_sort (std::shared_ptr< GraphType > graph)

  topology_sort2222.4

7.40 src/graph_algorithms/basic_graph/topology_sort/topologysort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "topologysort.h"
#include "../../basic_graph/graph_representation/graph_vertex/dfs_vertex.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
```

Classes

- class TopologySortTest

  TopologySortTest:

Namespaces

- anonymous_namespace{topologysort_test.h}

Functions

- TEST_F (TopologySortTest, test_topology_sort)

  test_topology_sort:topology_sort

Variables

- const int anonymous_namespace{topologysort_test.h}::TPS_N = 10

7.40.1 Function Documentation

7.40.1.1 TEST_F (**TopologySortTest** , test_topology_sort)

test_topology_sort:topology_sort

```
topology_sort
```

Definition at line 76 of file topologysort_test.h.

7.41 src/graph_algorithms/max_flow/ford_fulkerson/fordfulkerson.h File Reference

```
#include <memory>
#include <array>
#include <vector>
#include <functional>
#include "src/header.h"
#include "../../basic_graph/graph_bfs/bfs.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  std::shared_ptr< GraphType > IntroductionToAlgorithm::GraphAlgorithm::create_Gf (const std::shared_↩
  ptr< GraphType > graph, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >,
  GraphType::NUM > &flow)

create_Gf2626.2

- template<typename GraphType >
std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::ford_fulkerson (const std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

    ford_fulkersonford_fulkerson2626.2

7.42 src/graph_algorithms/max_flow/ford_fulkerson/fordfulkerson_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "fordfulkerson.h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph/graph.↩
h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph_↩
vertex/bfs_vertex.h"
```

Classes

- class FordFulkersonTest

    FordFulkersonTest:

Namespaces

- anonymous_namespace{fordfulkerson_test.h}

Functions

- TEST_F (FordFulkersonTest, test_ford_fulkerson)

    FordFulkersonTest: ford_fulkerson

Variables

- const int anonymous_namespace{fordfulkerson_test.h}::FF_N = 6

7.42.1 Function Documentation

7.42.1.1 TEST_F (**FordFulkersonTest** , test_ford_fulkerson)

FordFulkersonTest: ford_fulkerson

`test_ford_fulkerson` ford_fulkerson

Definition at line 72 of file fordfulkerson_test.h.

7.43 src/graph_algorithms/max_flow/generic_push_relabel/genericpushrelabel.h File Reference

```
#include <memory>
```

```
#include <array>
#include <vector>
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::push (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, typename GraphType::VIDType v_id, std::array< std::array< typename Graph↩
  Type::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  pushgeneric_push_relabelpush2626.4
- template<typename GraphType >
  GraphType::VIDType IntroductionToAlgorithm::GraphAlgorithm::min_v_at_Ef (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, const std::array< std::array< typename GraphType::E↩
  WeightType, GraphType::NUM >, GraphType::NUM > &flow)

  min_v_at_Efrelabelmin_v_at_Ef2626.4
- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::relabel (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, const std::array< std::array< typename GraphType::EWeightType, GraphType↩
  ::NUM >, GraphType::NUM > &flow)

  relabelgeneric_push_relabelrelabel2626.4
- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::initialize_preflow (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, std::array< std::array< typename GraphType::EWeightType, GraphType::↩
  NUM >, GraphType::NUM > &flow)

  initialize_preflowgeneric_push_relabel2626.4
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::generic_push_relabel (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  generic_push_relabel-2626.4

7.44 src/graph_algorithms/max_flow/generic_push_relabel/genericpushrelabel_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "genericpushrelabel.h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph/graph.↩
h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph_↩
vertex/flow_vertex.h"
```

Classes

- class GenericPushRelabelTest

  GenericPushRelabelTest:

Namespaces

- anonymous_namespace{genericpushrelabel_test.h}

Functions

- TEST_F (GenericPushRelabelTest, test_initialize_preflow)

  GenericPushRelabelTest: initialize_preflow
- TEST_F (GenericPushRelabelTest, test_min_v_at_Ef)

  GenericPushRelabelTest: min_v_at_Ef
- TEST_F (GenericPushRelabelTest, test_push)

  GenericPushRelabelTest: push
- TEST_F (GenericPushRelabelTest, test_relabel)

  GenericPushRelabelTest: relabel
- TEST_F (GenericPushRelabelTest, test_generic_push_relabel)

  GenericPushRelabelTest: generic_push_relabel

Variables

- const int anonymous_namespace{genericpushrelabel_test.h}::PR_N = 6

7.44.1 Function Documentation

7.44.1.1 TEST_F (**GenericPushRelabelTest** , test_initialize_preflow)

GenericPushRelabelTest: initialize_preflow

`test_initialize_preflow` initialize_preflow

Definition at line 77 of file genericpushrelabel_test.h.

7.44.1.2 TEST_F (**GenericPushRelabelTest** , test_min_v_at_Ef)

GenericPushRelabelTest: min_v_at_Ef

`test_min_v_at_Ef` min_v_at_Ef

Definition at line 110 of file genericpushrelabel_test.h.

7.44.1.3 TEST_F (**GenericPushRelabelTest** , test_push)

GenericPushRelabelTest: push

`test_push` push

Definition at line 125 of file genericpushrelabel_test.h.

7.44.1.4 TEST_F (GenericPushRelabelTest , test_relabel)

GenericPushRelabelTest: relabel

`test_relabel` relabel

Definition at line 164 of file genericpushrelabel_test.h.

7.44.1.5 TEST_F (GenericPushRelabelTest , test_generic_push_relabel)

GenericPushRelabelTest: generic_push_relabel

`test_generic_push_relabel` generic_push_relabel

Definition at line 197 of file genericpushrelabel_test.h.

7.45 src/graph_algorithms/max_flow/relabel_to_front/relabeltofront.h File Reference

```
#include <memory>
#include "../generic_push_relabel/genericpushrelabel.h"
#include "../../basic_graph/graph_representation/graph_vertex/front_flow_↩
vertex.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroduntionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::discharge (std::shared_ptr< GraphType > graph, typename GraphType::VIDType u_id, std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > &flow)

  discharge2626.5
- template<typename GraphType >
  List< ListNode< typename GraphType::VertexType > > IntroductionToAlgorithm::GraphAlgorithm::create↩_L (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VID↩Type dst)

  create_LL
- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::initial_vertex_NList (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  initial_vertex_NList
- template<typename GraphType >
  std::array< std::array< typename GraphType::EWeightType, GraphType::NUM >, GraphType::NUM > IntroductionToAlgorithm::GraphAlgorithm::relabel_to_front (std::shared_ptr< GraphType > graph, typename GraphType::VIDType src, typename GraphType::VIDType dst)

  relabel_to_front2626.5

7.46 src/graph_algorithms/max_flow/relabel_to_front/relabeltofront_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "relabeltofront.h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph/graph.↩
h"
```

Classes

- class RelabelToFrontTest

  RelabelToFrontTest:

Namespaces

- anonymous_namespace{relabeltofront_test.h}

Functions

- TEST_F (RelabelToFrontTest, test_relabel_to_front)

  RelabelToFrontTest: relabel_to_front

Variables

- const int anonymous_namespace{relabeltofront_test.h}::RTF_N = 6

7.46.1 Function Documentation

7.46.1.1 TEST_F (**RelabelToFrontTest** , test_relabel_to_front)

RelabelToFrontTest: relabel_to_front

`test_relabel_to_front` relabel_to_front

Definition at line 71 of file relabeltofront_test.h.

7.47 src/graph_algorithms/minimum_spanning_tree/kruskal/kruskal.h File Reference

```
#include "src/set_algorithms/disjoint_set/disjointset.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType,typename GraphType↩ ::VIDType)>>
  GraphType::EWeightType IntroductionToAlgorithm::GraphAlgorithm::kruskal (std::shared_ptr< GraphType > graph, ActionType pre_action=[ ](typename GraphType::VIDType, typename GraphType::VIDType){}, ActionType post_action=[ ](typename GraphType::VIDType, typename GraphType::VIDType){})

  kruskalKruskal2323.2

7.48 src/graph_algorithms/minimum_spanning_tree/kruskal/kruskal_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "kruskal.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "src/graph_algorithms/basic_graph/graph_representation/graph_↩
vertex/set_vertex.h"
```

Classes

- class KruskalTest

  KruskalTest:

Namespaces

- anonymous_namespace{kruskal_test.h}

Functions

- TEST_F (KruskalTest, test_kruskal)

  KruskalTest:kruskal

Variables

- const int anonymous_namespace{kruskal_test.h}::K_NUM =10

7.48.1 Function Documentation

7.48.1.1 TEST_F (**KruskalTest** , test_kruskal)

KruskalTest:kruskal

`test_kruskal`:kruskal

Definition at line 83 of file kruskal_test.h.

7.49 src/graph_algorithms/minimum_spanning_tree/prim/prim.h File Reference

```
#include <functional>
#include "src/queue_algorithms/min_queue/minqueue.h"
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType , typename ActionType = std::function< void(typename GraphType::VIDType)>>
  GraphType::EWeightType IntroductionToAlgorithm::GraphAlgorithm::prim (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id, ActionType pre_action=[ ](typename GraphType::VID←Type){}, ActionType post_action=[ ](typename GraphType::VIDType){})

  primPrim2323.2

7.50 src/graph_algorithms/minimum_spanning_tree/prim/prim_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "prim.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertexp.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
```

Classes

- class PrimTest

  PrimTest:

Namespaces

- anonymous_namespace{prim_test.h}

Functions

- TEST_F (PrimTest, test_prim)

  PrimTest:prim

Variables

- const int anonymous_namespace{prim_test.h}::PRIM_N = 10

7.50.1 Function Documentation

7.50.1.1 TEST_F (**PrimTest** , test_prim)

PrimTest:prim

`test_prim`:prim

Definition at line 81 of file prim_test.h.

7.51 src/graph_algorithms/single_source_shortest_path/bellman_ford/bellmanford.h File Reference

```
#include <memory>
#include "src/header.h"
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

    Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::initialize_single_source (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

    initialize_single_source2424.1
- template<typename VertexType >
  void IntroductionToAlgorithm::GraphAlgorithm::relax (std::shared_ptr< VertexType > from, std::shared_ptr< VertexType > to, typename VertexType::KeyType weight)

    relax2424.1
- template<typename GraphType >
  bool IntroductionToAlgorithm::GraphAlgorithm::bellman_ford (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

    bellman_fordbellman_ford2424.1

7.52 src/graph_algorithms/single_source_shortest_path/bellman_ford/bellmanford_↩ test.h File Reference

```
#include "src/google_test/gtest.h"
#include "bellmanford.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertexp.h"
```

Classes

- class BellmanFordTest

    BellmanFordTest:

Namespaces

- anonymous_namespace{bellmanford_test.h}

Functions

- TEST_F (BellmanFordTest, test_initialize_single_source)

  BellmanFordTest:initialize_single_source
- TEST_F (BellmanFordTest, test_relax)

  BellmanFordTest:relax
- TEST_F (BellmanFordTest, test_bellman_ford)

  BellmanFordTest:bellman_ford

Variables

- const int anonymous_namespace{bellmanford_test.h}::B_NUM =10

7.52.1 Function Documentation

7.52.1.1 TEST_F (**BellmanFordTest** , test_initialize_single_source)

BellmanFordTest:initialize_single_source

`test_initialize_single_source`:initialize_single_source

Definition at line 84 of file bellmanford_test.h.

7.52.1.2 TEST_F (**BellmanFordTest** , test_relax)

BellmanFordTest:relax

`test_relax`:relax

Definition at line 102 of file bellmanford_test.h.

7.52.1.3 TEST_F (**BellmanFordTest** , test_bellman_ford)

BellmanFordTest:bellman_ford

`test_bellman_ford`:bellman_ford

Definition at line 116 of file bellmanford_test.h.

7.53 src/graph_algorithms/single_source_shortest_path/dag_shortest_path/dagshortpath.h File Reference

```
#include <memory>
#include <functional>
#include "../bellman_ford/bellmanford.h"
#include "../../basic_graph/topology_sort/topologysort.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::dag_shortest_path (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

  dag_shortest_pathdag_shortest_path2424.2

7.54 src/graph_algorithms/single_source_shortest_path/dag_shortest_path/dagshortpath↩_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "dagshortpath.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "../../basic_graph/graph_representation/graph_vertex/dfs_vertex.h"
```

Classes

- class DagShortestPathTest

  DagShortestPathTest:

Namespaces

- anonymous_namespace{dagshortpath_test.h}

Functions

- TEST_F (DagShortestPathTest, test_dag_shortest_path)

  DagShortestPathTest:dag_shortest_path

Variables

- const int anonymous_namespace{dagshortpath_test.h}::DSP_NUM =10

7.54.1 Function Documentation

7.54.1.1 TEST_F (DagShortestPathTest , test_dag_shortest_path)

DagShortestPathTest:dag_shortest_path

`test_dag_shortest_path`:dag_shortest_path

Definition at line 74 of file dagshortpath_test.h.

7.55 src/graph_algorithms/single_source_shortest_path/dijkstra/dijkstra.h File Reference

```
#include <vector>
#include "../bellman_ford/bellmanford.h"
#include "src/queue_algorithms/min_queue/minqueue.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm

  Namespace of GraphAlgorithm.

Functions

- template<typename GraphType >
  void IntroductionToAlgorithm::GraphAlgorithm::dijkstra (std::shared_ptr< GraphType > graph, typename GraphType::VIDType source_id)

  dijkstradijkstra2424.3

7.56 src/graph_algorithms/single_source_shortest_path/dijkstra/dijkstra_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "dijkstra.h"
#include "../../basic_graph/graph_representation/graph/graph.h"
#include "../../basic_graph/graph_representation/graph_vertex/vertexp.h"
```

Classes

- class DijkstraTest

  DijkstraTest:

Namespaces

- anonymous_namespace{dijkstra_test.h}

Functions

- TEST_F (DijkstraTest, test_dijkstra)

  DijkstraTest:dijkstra

Variables

- const int anonymous_namespace{dijkstra_test.h}::DIJK_NUM =10

7.56.1 Function Documentation

7.56.1.1 TEST_F (DijkstraTest , test_dijkstra)

DijkstraTest:dijkstra

`test_dijkstra`:dijkstra

Definition at line 75 of file dijkstra_test.h.

7.57 src/header.h File Reference

```
#include <algorithm>
#include <memory>
#include <vector>
#include <ostream>
```

Namespaces

- IntroductionToAlgorithm
  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm
  Namespace of SortAlgorithm.
- IntroductionToAlgorithm::SelectAlgorithm
  Namespace of SelectAlgorithm.
- IntroductionToAlgorithm::DynamicProgrammingAlgorithm
  Namespace of DynamicProgrammingAlgorithm.
- IntroductionToAlgorithm::TreeAlgorithm
  Namespace of TreeAlgorithm.
- IntroductionToAlgorithm::QueueAlgorithm
  Namespace of QueueAlgorithm.
- IntroductionToAlgorithm::SetAlgorithm
  Namespace of SetAlgorithm.
- IntroductionToAlgorithm::GraphAlgorithm
  Namespace of GraphAlgorithm.
- IntroductionToAlgorithm::StringMatchingAlgorithm
  Namespace of StringMatchingAlgorithm.

Functions

- template<typename T >
  T IntroductionToAlgorithm::GraphAlgorithm::unlimit ()
  unlimit
- template<typename T >
  bool IntroductionToAlgorithm::GraphAlgorithm::is_unlimit (T t)
  is_unlimit
- template<typename VertexType >
  std::vector< typename VertexType::VIDType > IntroductionToAlgorithm::GraphAlgorithm::get_path (const std::shared_ptr< VertexType > v_from, const std::shared_ptr< VertexType > v_to)
  get_path
- template<typename MatrixType >
  std::string IntroductionToAlgorithm::GraphAlgorithm::matrix_string (const MatrixType &matrix)
  matrix_string

7.58 src/queue_algorithms/min_queue/minqueue.h File Reference

```
#include <vector>
#include <memory>
#include <functional>
#include "src/header.h"
```

Classes

- class IntroductionToAlgorithm::QueueAlgorithm::MinQueue< T, TKeyType >

  MinQueue66.5

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::QueueAlgorithm

  Namespace of QueueAlgorithm.

7.59 src/queue_algorithms/min_queue/minqueue_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "minqueue.h"
```

Classes

- struct Node

  Node:
- class MinQueueTest

  MinQueueTest:

Namespaces

- anonymous_namespace{minqueue_test.h}

Functions

- TEST_F (MinQueueTest, test_min)

  MinQueueTest:
- TEST_F (MinQueueTest, test_extract_min)

  MinQueueTest:
- TEST_F (MinQueueTest, test_insert)

  MinQueueTest:
- TEST_F (MinQueueTest, test_is_empty)

  MinQueueTest:
- TEST_F (MinQueueTest, test_index_inqueue)

  MinQueueTest:

- TEST_F (MinQueueTest, test_decreate_key)

  MinQueueTest:
- TEST_F (MinQueueTest, test_setupHeap)

  MinQueueTest:
- TEST_F (MinQueueTest, test_heapify)

  MinQueueTest:

Variables

- const int anonymous_namespace{minqueue_test.h}::Q_NUM =10

7.59.1 Function Documentation

7.59.1.1 TEST_F (**MinQueueTest** , test_min)

MinQueueTest:

`test_minmin()`

Definition at line 72 of file minqueue_test.h.

7.59.1.2 TEST_F (**MinQueueTest** , test_extract_min)

MinQueueTest:

`test_extract_minextract_min()`

Definition at line 82 of file minqueue_test.h.

7.59.1.3 TEST_F (**MinQueueTest** , test_insert)

MinQueueTest:

`test_insertinsert(...)`

Definition at line 92 of file minqueue_test.h.

7.59.1.4 TEST_F (**MinQueueTest** , test_is_empty)

MinQueueTest:

`test_is_emptyis_empty()`

Definition at line 121 of file minqueue_test.h.

7.59.1.5 TEST_F (**MinQueueTest** , test_index_inqueue)

MinQueueTest:

`test_index_inqueueindex_inqueue(...)`

Definition at line 135 of file minqueue_test.h.

7.59.1.6 TEST_F (**MinQueueTest** , test_decreate_key)

MinQueueTest:

`test_decreate_keydecreate_key(...)test_inserttest_decreate_key`

Definition at line 164 of file minqueue_test.h.

7.59.1.7 TEST_F (MinQueueTest , test_setupHeap)

MinQueueTest:

`test_setupHeapsetupHeap()`

Definition at line 173 of file minqueue_test.h.

7.59.1.8 TEST_F (MinQueueTest , test_heapify)

MinQueueTest:

`test_heapifyheapify(...)extract_mintest_heapify`

Definition at line 182 of file minqueue_test.h.

7.60 src/select_algorithms/good_select/goodselect.h File Reference

```
#include <vector>
#include "src/sort_algorithms/quick_sort/quicksort.h"
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SelectAlgorithm

  Namespace of SelectAlgorithm.

Functions

- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>> std::iterator_traits< Iterator >::value_type IntroductionToAlgorithm::SelectAlgorithm::good_select (const Iterator begin, const Iterator end, typename std::iterator_traits< Iterator >::difference_type rank, CompareType compare=CompareType())

  good_select 99.3 O(n)

7.61 src/select_algorithms/good_select/goodselect_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "goodselect.h"
```

Functions

- TEST (test_good_select, test_C_array)

  good_select_testC

7.61.1 Function Documentation

7.61.1.1 TEST (test_good_select , test_C_array)

good_select_testC

Definition at line 29 of file goodselect_test.h.

7.62 src/select_algorithms/randomized_select/randomizedselect.h File Reference

```
#include <src/sort_algorithms/quick_sort/quicksort.h>
#include <random>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SelectAlgorithm

  Namespace of SelectAlgorithm.

Functions

- template<typename IntType >
  IntType IntroductionToAlgorithm::SelectAlgorithm::radom_index (IntType begin, IntType end)

  radom_index
- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  std::iterator_traits< Iterator >::value_type IntroductionToAlgorithm::SelectAlgorithm::randomized_select (const Iterator begin, const Iterator end, typename std::iterator_traits< Iterator >::difference_type rank, CompareType compare=CompareType())

  randomized_select 99.2

7.63 src/select_algorithms/randomized_select/randomizedselect_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "randomizedselect.h"
```

Functions

- TEST (test_radom_index, test_random)

  radom_index_test
- TEST (test_randomized_select, test_C_array)

  randomized_select_testC

7.63.1 Function Documentation

7.63.1.1 TEST (test_radom_index , test_random)

radom_index_test

10

Definition at line 12 of file randomizedselect_test.h.

7.63.1.2 TEST (test_randomized_select , test_C_array)

randomized_select_testC

Definition at line 31 of file randomizedselect_test.h.

7.64 src/set_algorithms/disjoint_set/disjointset.h File Reference

```
#include <memory>
#include <vector>
```

Classes

- struct IntroductionToAlgorithm::SetAlgorithm::DisjointSetNode< KType >

  DisjointSetNode2121.3

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SetAlgorithm

  Namespace of SetAlgorithm.

7.65 src/set_algorithms/disjoint_set/disjointset_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "disjointset.h"
```

Classes

- class DisjointSetNodeTest

  DisjointSetNodeTest:

Namespaces

- anonymous_namespace{disjointset_test.h}

Functions

- TEST_F (DisjointSetNodeTest, test_make_set)

  DisjointSetNodeTest:DisjointSetNodeTest.
- TEST_F (DisjointSetNodeTest, test_find_set)

  DisjointSetNodeTest:DisjointSetNodeTest.
- TEST_F (DisjointSetNodeTest, test_link)

  DisjointSetNodeTest:DisjointSetNodeTest.
- TEST_F (DisjointSetNodeTest, test_union)

  DisjointSetNodeTest:DisjointSetNodeTest.

Variables

- const int anonymous_namespace{disjointset_test.h}::S_NUM =20

7.65.1 Function Documentation

7.65.1.1 TEST_F (**DisjointSetNodeTest** , test_make_set)

DisjointSetNodeTest:DisjointSetNodeTest.

`test_make_set`: make_set

Definition at line 53 of file disjointset_test.h.

7.65.1.2 TEST_F (**DisjointSetNodeTest** , test_find_set)

DisjointSetNodeTest:DisjointSetNodeTest.

`test_find_set`: find_set

Definition at line 68 of file disjointset_test.h.

7.65.1.3 TEST_F (**DisjointSetNodeTest** , test_link)

DisjointSetNodeTest:DisjointSetNodeTest.

`test_link`: link

Definition at line 82 of file disjointset_test.h.

7.65.1.4 TEST_F (**DisjointSetNodeTest** , test_union)

DisjointSetNodeTest:DisjointSetNodeTest.

`test_union`: union

Definition at line 112 of file disjointset_test.h.

7.66 src/sort_algorithms/bucket_sort/bucketsort.h File Reference

```
#include "../quick_sort/quicksort.h"
#include <vector>
#include <cassert>
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

    Namespace of SortAlgorithm.

Functions

- template<typename Iterator >
  void IntroductionToAlgorithm::SortAlgorithm::bucket_sort (const Iterator begin, const Iterator end, const typename std::iterator_traits< Iterator >::value_type &min_val, const typename std::iterator_traits< Iterator >↩
  ::value_type &max_val)

    bucket_sort8 8.4

7.67 src/sort_algorithms/bucket_sort/bucketsort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "bucketsort.h"
```

Functions

- TEST (test_bucket_sort, test_C_array)

    bucket_sort_testC
- TEST (test_bucket_sort, test_std_container)

    bucket_sort_teststd::array

7.67.1 Function Documentation

7.67.1.1 TEST (test_bucket_sort , test_C_array)

bucket_sort_testC

std::sort()

Definition at line 30 of file bucketsort_test.h.

7.67.1.2 TEST (test_bucket_sort , test_std_container)

bucket_sort_teststd::array

std::array std::array std::sort()

Definition at line 61 of file bucketsort_test.h.

7.68 src/sort_algorithms/count_sort/countsort.h File Reference

```
#include <vector>
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

    Namespace of SortAlgorithm.

Functions

- template<typename Iterator >
  void IntroductionToAlgorithm::SortAlgorithm::count_sort (const Iterator begin, const Iterator end, const typename std::iterator_traits< Iterator >::value_type &max_val)

    count_sort8 8.2

7.69 src/sort_algorithms/count_sort/countsort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "countsort.h"
```

Functions

- TEST (test_count_sort, test_C_array)

    count_sort_testC
- TEST (test_count_sort, test_std_container)

    count_sort_teststd::array

7.69.1 Function Documentation

7.69.1.1 TEST (test_count_sort , test_C_array)

count_sort_testC

std::sort()

Definition at line 30 of file countsort_test.h.

7.69.1.2 TEST (test_count_sort , test_std_container)

count_sort_teststd::array

std::array std::array std::sort()

Definition at line 61 of file countsort_test.h.

7.70 src/sort_algorithms/heap_sort/heapsort.h File Reference

Classes

- class IntroductionToAlgorithm::SortAlgorithm::Sort_Heap< Iterator, CompareType >

    Sort_Heap6

Namespaces

- IntroductionToAlgorithm

    Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

    Namespace of SortAlgorithm.

7.71 src/sort_algorithms/heap_sort/heapsort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "heapsort.h"
```

Functions

- TEST (test_heap_sort, test_C_array)

    heap_sort_testC
- TEST (test_heap_sort, test_std_container)

    heap_sort_teststd::array

7.71.1 Function Documentation

7.71.1.1 TEST (test_heap_sort , test_C_array)

heap_sort_testC

std::sort()

Definition at line 30 of file heapsort_test.h.

7.71.1.2 TEST (test_heap_sort , test_std_container)

heap_sort_teststd::array

std::array std::array std::sort()

Definition at line 62 of file heapsort_test.h.

7.72 src/sort_algorithms/insert_sort/insertsort.h File Reference

Namespaces

- IntroductionToAlgorithm

    Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

    Namespace of SortAlgorithm.

Functions

- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  void IntroductionToAlgorithm::SortAlgorithm::insert_sort (const Iterator begin, const Iterator end, Compare↩
  Type compare=CompareType())

    insert_sort 2.1

7.73 src/sort_algorithms/insert_sort/insertsort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "insertsort.h"
```

Functions

- TEST (test_insert_sort, test_C_array)

    insert_sort_testC
- TEST (test_insert_sort, test_std_container)

    insert_sort_teststd::array

7.73.1 Function Documentation

7.73.1.1 TEST (test_insert_sort , test_C_array)

insert_sort_testC

std::sort()

Definition at line 30 of file insertsort_test.h.

7.73.1.2 TEST (test_insert_sort , test_std_container)

insert_sort_teststd::array

std::array std::array std::sort()

Definition at line 61 of file insertsort_test.h.

7.74 src/sort_algorithms/merge_sort/mergesort.h File Reference

```
#include <vector>
```

Namespaces

- IntroductionToAlgorithm

    Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

    Namespace of SortAlgorithm.

Functions

- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  void IntroductionToAlgorithm::SortAlgorithm::merge (const Iterator begin, const Iterator end, const Iterator middle, CompareType compare=CompareType())

    merge 2.3.1
- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  void IntroductionToAlgorithm::SortAlgorithm::merge_sort (const Iterator begin, const Iterator end, Compare↩ Type compare=CompareType())

    merge_sort 2.3.1

7.75 src/sort_algorithms/merge_sort/mergesort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "mergesort.h"
```

Functions

- TEST (test_merge_sort, test_C_array)

  merge_sort_testC
- TEST (test_merge_sort, test_std_container)

  merge_sort_teststd::array
- TEST (test_merge, test_C_array)

  merge_testC
- TEST (test_merge, test_std_container)

  merge_teststd::array

7.75.1 Function Documentation

7.75.1.1 TEST (test_merge_sort , test_C_array)

merge_sort_testC

std::sort()

Definition at line 31 of file mergesort_test.h.

7.75.1.2 TEST (test_merge_sort , test_std_container)

merge_sort_teststd::array

std::array std::array std::sort()

Definition at line 62 of file mergesort_test.h.

7.75.1.3 TEST (test_merge , test_C_array)

merge_testC

1100 std::sort()

Definition at line 92 of file mergesort_test.h.

7.75.1.4 TEST (test_merge , test_std_container)

merge_teststd::array

std::array 1100 std::sort()

Definition at line 128 of file mergesort_test.h.

7.76 src/sort_algorithms/quick_sort/quicksort.h File Reference

```
#include <assert.h>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

  Namespace of SortAlgorithm.

Functions

- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  Iterator IntroductionToAlgorithm::SortAlgorithm::partition (const Iterator begin, const Iterator end, const Iterator partition_iter, CompareType compare=CompareType())

  partition 7
- template<typename Iterator , typename CompareType = std::less<typename std::iterator_traits<Iterator>::value_type>>
  void IntroductionToAlgorithm::SortAlgorithm::quick_sort (const Iterator begin, const Iterator end, Compare↩Type compare=CompareType())

  quick_sort 7

7.77 src/sort_algorithms/quick_sort/quicksort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "quicksort.h"
```

Functions

- TEST (test_partition, test_C_array)

  partition_testC
- TEST (test_partition, test_std_container)

  partition_teststd::array
- TEST (test_quick_sort, test_C_array)

  quick_sort_testC
- TEST (test_quick_sort, test_std_container)

  quick_sort_teststd::array

7.77.1 Function Documentation

7.77.1.1 TEST (test_partition , test_C_array)

partition_testC

Definition at line 30 of file quicksort_test.h.

7.77.1.2 TEST (test_partition , test_std_container)

partition_teststd::array

std::array std::array

Definition at line 115 of file quicksort_test.h.

7.77.1.3 TEST (test_quick_sort , test_C_array)

quick_sort_testC

std::sort()

Definition at line 145 of file quicksort_test.h.

7.77.1.4 TEST (test_quick_sort , test_std_container)

quick_sort_teststd::array

std::array std::array std::sort()

Definition at line 176 of file quicksort_test.h.

7.78 src/sort_algorithms/radix_sort/radixsort.h File Reference

```
#include "../insert_sort/insertsort.h"
#include <cmath>
#include <iostream>
#include <cassert>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::SortAlgorithm

  Namespace of SortAlgorithm.

Functions

- template<typename T >
  T IntroductionToAlgorithm::SortAlgorithm::digi_on_N (T num, std::size_t n)

  digi_on_N
- template<typename Iterator >
  void IntroductionToAlgorithm::SortAlgorithm::radix_sort (const Iterator begin, const Iterator end, std::size_t radix_width)

  radix_sort8 8.3

7.79 src/sort_algorithms/radix_sort/radixsort_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "radixsort.h"
```

Functions

- TEST (test_radix_sort, test_digi_on_N)

  radix_sort_testtest_digi_on_N:0
- TEST (test_radix_sort, test_C_array)

radix_sort_testC
- TEST (test_radix_sort, test_std_container)

  radix_sort_teststd::array

7.79.1 Function Documentation

7.79.1.1 TEST (test_radix_sort , test_digi_on_N)

radix_sort_testtest_digi_on_N:0

123456789

Definition at line 30 of file radixsort_test.h.

7.79.1.2 TEST (test_radix_sort , test_C_array)

radix_sort_testC

std::sort()

Definition at line 42 of file radixsort_test.h.

7.79.1.3 TEST (test_radix_sort , test_std_container)

radix_sort_teststd::array

std::array std::array std::sort()

Definition at line 73 of file radixsort_test.h.

7.80 src/string_matching_algorithms/finite_automaton_match/finiteautomatonmatch.h File Reference

```
#include <vector>
#include <ostream>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::StringMatchingAlgorithm

  Namespace of StringMatchingAlgorithm.

Functions

- template<typename Iterator >
  std::iterator_traits< Iterator >::difference_type IntroductionToAlgorithm::StringMatchingAlgorithm::index_↩
  of_M (Iterator beginM, Iterator endM, typename std::iterator_traits< Iterator >::value_type a)

  index_of_M a3232.3
- template<typename Iterator >
  bool IntroductionToAlgorithm::StringMatchingAlgorithm::is_end_with (Iterator begin, Iterator k_iter, Iterator q_iter, typename std::iterator_traits< Iterator >::value_type a)

  is_end_with Pk(Pq a)3232.3

- template<typename PIterator , typename MIterator >
  void IntroductionToAlgorithm::StringMatchingAlgorithm::get_delta (const PIterator P_begin, const PIterator P_end, const MIterator M_begin, const MIterator M_end, std::vector< std::vector< int >> &delta)

  get_delt 3232.3
- template<typename IteratorT , typename IteratorP , typename IteratorM >
  std::vector< int > IntroductionToAlgorithm::StringMatchingAlgorithm::finite_automaton_match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end, const IteratorM iterM_begin, const IteratorM iterM_end)

  finite_automaton_match 3232.3

7.81 src/string_matching_algorithms/finite_automaton_match/finiteautomatonmatch_↩ test.h File Reference

```
#include "src/google_test/gtest.h"
#include "finiteautomatonmatch.h"
```

Functions

- TEST (MatchTest, test_index_of_M)

  test_index_of_M index_of_M
- TEST (MatchTest, test_is_end_with)

  test_is_end_with is_end_with
- TEST (MatchTest, test_get_delta)

  test_get_delta get_delta
- TEST (MatchTest, test_finite_automaton_match)

  test_finite_automaton_match

7.81.1 Function Documentation

7.81.1.1 TEST (MatchTest , test_index_of_M)

test_index_of_M index_of_M

Definition at line 29 of file finiteautomatonmatch_test.h.

7.81.1.2 TEST (MatchTest , test_is_end_with)

test_is_end_with is_end_with

Definition at line 47 of file finiteautomatonmatch_test.h.

7.81.1.3 TEST (MatchTest , test_get_delta)

test_get_delta get_delta

Definition at line 61 of file finiteautomatonmatch_test.h.

7.81.1.4 TEST (MatchTest , test_finite_automaton_match)

test_finite_automaton_match

TTT

Definition at line 81 of file finiteautomatonmatch_test.h.

7.82 src/string_matching_algorithms/kmp_match/kmp.h File Reference

```
#include <vector>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::StringMatchingAlgorithm

  Namespace of StringMatchingAlgorithm.

Functions

- template<typename IteratorP >
  std::vector< int > IntroductionToAlgorithm::StringMatchingAlgorithm::get_pai (const IteratorP iterP_begin, const IteratorP iterP_end)

  get_pai KMP3232.4
- template<typename IteratorT , typename IteratorP >
  std::vector< int > IntroductionToAlgorithm::StringMatchingAlgorithm::kmp_match (const IteratorT iterT_↩
  begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end)

  kmp_match KMP3232.4

7.83 src/string_matching_algorithms/kmp_match/kmp_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "kmp.h"
```

Functions

- TEST (MatchTest, test_get_pai)

  test_get_pai get_pai
- TEST (MatchTest, test_kmp_match)

  test_kmp_matchKMP

7.83.1 Function Documentation

7.83.1.1 TEST (MatchTest , test_get_pai)

test_get_pai get_pai

Definition at line 27 of file kmp_test.h.

7.83.1.2 TEST (MatchTest , test_kmp_match)

test_kmp_matchKMP

TTT

Definition at line 40 of file kmp_test.h.

7.84 src/string_matching_algorithms/rabin_karp_match/rabinkarpmatch.h File Reference

```
#include <type_traits>
#include <vector>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::StringMatchingAlgorithm

  Namespace of StringMatchingAlgorithm.

Functions

- template<typename T >
  T IntroductionToAlgorithm::StringMatchingAlgorithm::get_h (T radix_d, T len_m, T mod_q)

  get_h rabin_karp get_h 3232.2
- template<typename IteratorT , typename IteratorP >
  std::vector< int > IntroductionToAlgorithm::StringMatchingAlgorithm::rabin_karp_match (const Iterator↩T iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end, unsigned radix_d, unsigned mod_q)

  rabin_karp_match rabin_karp3232.2

7.85 src/string_matching_algorithms/rabin_karp_match/rabinkarpmatch_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "rabinkarpmatch.h"
```

Functions

- TEST (MatchTest, test_get_h)

  test_get_h get_h
- TEST (MatchTest, test_rabin_karp_match)

  test_rabin_karp_matchrabin_karp_match

7.85.1 Function Documentation

7.85.1.1 TEST (MatchTest , test_get_h)

test_get_h get_h

Definition at line 28 of file rabinkarpmatch_test.h.

7.85.1.2 TEST (MatchTest , test_rabin_karp_match)

test_rabin_karp_matchrabin_karp_match

TT

Definition at line 42 of file rabinkarpmatch_test.h.

7.86 src/string_matching_algorithms/regular_match/match.h File Reference

```
#include <vector>
```

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::StringMatchingAlgorithm

  Namespace of StringMatchingAlgorithm.

Functions

- template<typename IteratorT , typename IteratorP >
  std::vector< int > IntroductionToAlgorithm::StringMatchingAlgorithm::match (const IteratorT iterT_begin, const IteratorT iterT_end, const IteratorP iterP_begin, const IteratorP iterP_end)

  match 32,32.1

7.87 src/string_matching_algorithms/regular_match/match_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "match.h"
```

Functions

- TEST (MatchTest, test_regular_match)

  test_regular_match

7.87.1 Function Documentation

7.87.1.1 TEST (MatchTest , test_regular_match)

test_regular_match

TT

Definition at line 31 of file match_test.h.

7.88 src/tree_algorithms/binarytree/binarytree.h File Reference

```
#include <memory>
#include <functional>
```

Classes

- struct IntroductionToAlgorithm::TreeAlgorithm::BinaryTree< NodeT >

  BinaryTree1010.4

Namespaces

- IntroductionToAlgorithm

  Namespace of IntroductionToAlgorithm.
- IntroductionToAlgorithm::TreeAlgorithm

  Namespace of TreeAlgorithm.

Functions

- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void IntroductionToAlgorithm::TreeAlgorithm::inorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  inorder_walk
- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void IntroductionToAlgorithm::TreeAlgorithm::preorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  preorder_walk
- template<typename NodeType , typename ActionType = std::function<void (typename NodeType::T)>>
  void IntroductionToAlgorithm::TreeAlgorithm::postorder_walk (std::shared_ptr< NodeType > root, ActionType action=[ ](typename NodeType::T){})

  postorder_walk
- template<typename NodeType >
  void IntroductionToAlgorithm::TreeAlgorithm::left_rotate (std::shared_ptr< NodeType > node, std::shared_↩
  ptr< NodeType > &root)

  left_rotate
- template<typename NodeType >
  void IntroductionToAlgorithm::TreeAlgorithm::right_rotate (std::shared_ptr< NodeType > node, std::shared↩
  _ptr< NodeType > &root)

  right_rotate
- template<typename NodeType >
  void IntroductionToAlgorithm::TreeAlgorithm::transplant (std::shared_ptr< NodeType > node_src, std↩
  ::shared_ptr< NodeType >node_dst, std::shared_ptr< NodeType > &root)

  transplant

7.89 src/tree_algorithms/binarytree/binarytree_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "binarytree.h"
#include "../binarytreenode/binarytreenode.h"
#include <sstream>
```

Classes

- class BinaryTreeTest

  BinaryTreeTest:

Functions

- TEST_F (BinaryTreeTest, test_tree)

  test_tree:
- TEST_F (BinaryTreeTest, test_inorder_walk)

  test_inorder_walk:
- TEST_F (BinaryTreeTest, test_preorder_walk)

  test_preorder_walk:
- TEST_F (BinaryTreeTest, test_postorder_walk)

  test_postorder_walk:
- TEST_F (BinaryTreeTest, test_left_rotate)

  test_left_rotate:
- TEST_F (BinaryTreeTest, test_right_rotate)

  test_right_rotate:
- TEST_F (BinaryTreeTest, test_right_transplant)

  test_right_transplant:

7.89.1 Function Documentation

7.89.1.1 TEST_F (BinaryTreeTest , test_tree)

test_tree:

Definition at line 96 of file binarytree_test.h.

7.89.1.2 TEST_F (BinaryTreeTest , test_inorder_walk)

test_inorder_walk:

Definition at line 106 of file binarytree_test.h.

7.89.1.3 TEST_F (BinaryTreeTest , test_preorder_walk)

test_preorder_walk:

Definition at line 129 of file binarytree_test.h.

7.89.1.4 TEST_F (BinaryTreeTest , test_postorder_walk)

test_postorder_walk:

Definition at line 152 of file binarytree_test.h.

7.89.1.5 TEST_F (BinaryTreeTest , test_left_rotate)

test_left_rotate:

Definition at line 175 of file binarytree_test.h.

7.89.1.6 TEST_F (BinaryTreeTest , test_right_rotate)

test_right_rotate:

Definition at line 207 of file binarytree_test.h.

7.89.1.7 TEST_F (BinaryTreeTest , test_right_transplant)

test_right_transplant:

Definition at line 238 of file binarytree_test.h.

7.90 src/tree_algorithms/binarytreenode/binarytreenode.h File Reference

```
#include <memory>
#include <string>
#include <sstream>
```

Classes

- struct IntroductionToAlgorithm::TreeAlgorithm::BinaryTreeNode< KType >

  BinaryTreeNodexxxx

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.

- IntroductionToAlgorithm::TreeAlgorithm

  Namespace of TreeAlgorithm.

7.91 src/tree_algorithms/binarytreenode/binarytreenode_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "binarytreenode.h"
```

Classes

- class BinaryTreeNodeTest
  BinaryTreeNodeTest:

Functions

- TEST_F (BinaryTreeNodeTest, test_default_node)
  binary_tree_node_test
- TEST_F (BinaryTreeNodeTest, test_to_string)
  binary_tree_node_test
- TEST_F (BinaryTreeNodeTest, test_to_xml)
  binary_tree_node_test
- TEST_F (BinaryTreeNodeTest, test_is_left_child)
  binary_tree_node_test
- TEST_F (BinaryTreeNodeTest, test_is_right_child)
  binary_tree_node_test

7.91.1 Function Documentation

7.91.1.1 TEST_F (BinaryTreeNodeTest , test_default_node)

binary_tree_node_test

Definition at line 55 of file binarytreenode_test.h.

7.91.1.2 TEST_F (BinaryTreeNodeTest , test_to_string)

binary_tree_node_test

```
to_string()
```

Definition at line 67 of file binarytreenode_test.h.

7.91.1.3 TEST_F (BinaryTreeNodeTest , test_to_xml)

binary_tree_node_test

```
to_xml()
```

Definition at line 77 of file binarytreenode_test.h.

7.91.1.4 TEST_F (BinaryTreeNodeTest , test_is_left_child)

binary_tree_node_test

```
 is_left_child()
```

Definition at line 87 of file binarytreenode_test.h.

7.91.1.5 TEST_F (BinaryTreeNodeTest , test_is_right_child)

binary_tree_node_test

```
is_right_child()
```

Definition at line 98 of file binarytreenode_test.h.

7.92 src/tree_algorithms/searchtree/searchtree.h File Reference

```
#include "../binarytree/binarytree.h"
```

Classes

- class IntroductionToAlgorithm::TreeAlgorithm::SearchTree< NodeType >

  SearchTree12

Namespaces

- IntroductionToAlgorithm

  Namespace of IntrodunctionToAlgorithm.
- IntroductionToAlgorithm::TreeAlgorithm

  Namespace of TreeAlgorithm.

7.93 src/tree_algorithms/searchtree/searchtree_test.h File Reference

```
#include "src/google_test/gtest.h"
#include "searchtree.h"
#include "../binarytreenode/binarytreenode.h"
#include "../binarytree/binarytree.h"
#include <sstream>
```

Classes

- class SearchTreeTest

  SearchTreeTest:

Namespaces

- anonymous_namespace{searchtree_test.h}

Functions

- TEST_F (SearchTreeTest, search_test)

  search_test:
- TEST_F (SearchTreeTest, min_test)

  min_test:
- TEST_F (SearchTreeTest, max_test)

max_test:
- TEST_F (SearchTreeTest, predecesor_test)

  predecesor_test:
- TEST_F (SearchTreeTest, successor_test)

  successor_test:
- TEST_F (SearchTreeTest, insert_test)

  successor_test:
- TEST_F (SearchTreeTest, remove_test)

  successor_test:

Variables

- const int anonymous_namespace{searchtree_test.h}::NODE_NUM =9

7.93.1 Function Documentation

7.93.1.1 TEST_F (SearchTreeTest , search_test)

search_test:

Definition at line 108 of file searchtree_test.h.

7.93.1.2 TEST_F (SearchTreeTest , min_test)

min_test:

Definition at line 130 of file searchtree_test.h.

7.93.1.3 TEST_F (SearchTreeTest , max_test)

max_test:

Definition at line 145 of file searchtree_test.h.

7.93.1.4 TEST_F (SearchTreeTest , predecesor_test)

predecesor_test:

Definition at line 160 of file searchtree_test.h.

7.93.1.5 TEST_F (SearchTreeTest , successor_test)

successor_test:

Definition at line 179 of file searchtree_test.h.

7.93.1.6 TEST_F (SearchTreeTest , insert_test)

successor_test:

Definition at line 199 of file searchtree_test.h.

7.93.1.7 TEST_F (SearchTreeTest , remove_test)

successor_test:

Definition at line 267 of file searchtree_test.h.