컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 40강 - 다익스트라의 최단 경로

Fast campus

학습 목표

다익스트라의 최단 경로

1) 다익스트라의 최단 경로 알고리즘의 원리와 구현 방법에 대해서 이해할 수 있습니다.
2) 다익스트라의 최단 경로 알고리즘을 우선순위 큐를 이용해 C 언어로 구현할 수 있습니다.

다익스트라의 최단 경로

다익스트라의 최단 경로

1) 다익스트라의 최단 경로(Dijkstra's Shortest Path Algorithm)는 매우 잘 알려진 알고리즘입니다.
2) 각 간선에 대한 정보를 우선순위 큐에 담아 처리하는 방식으로 동작 원리가 프림 알고리즘과 흡사합니다.
3) 현실 세계에서는 음의 간선이 존재하지 않기 때문에 다익스트라는 현실 세계에 적합한 알고리즘입니다.

다익스트라의 최단 경로

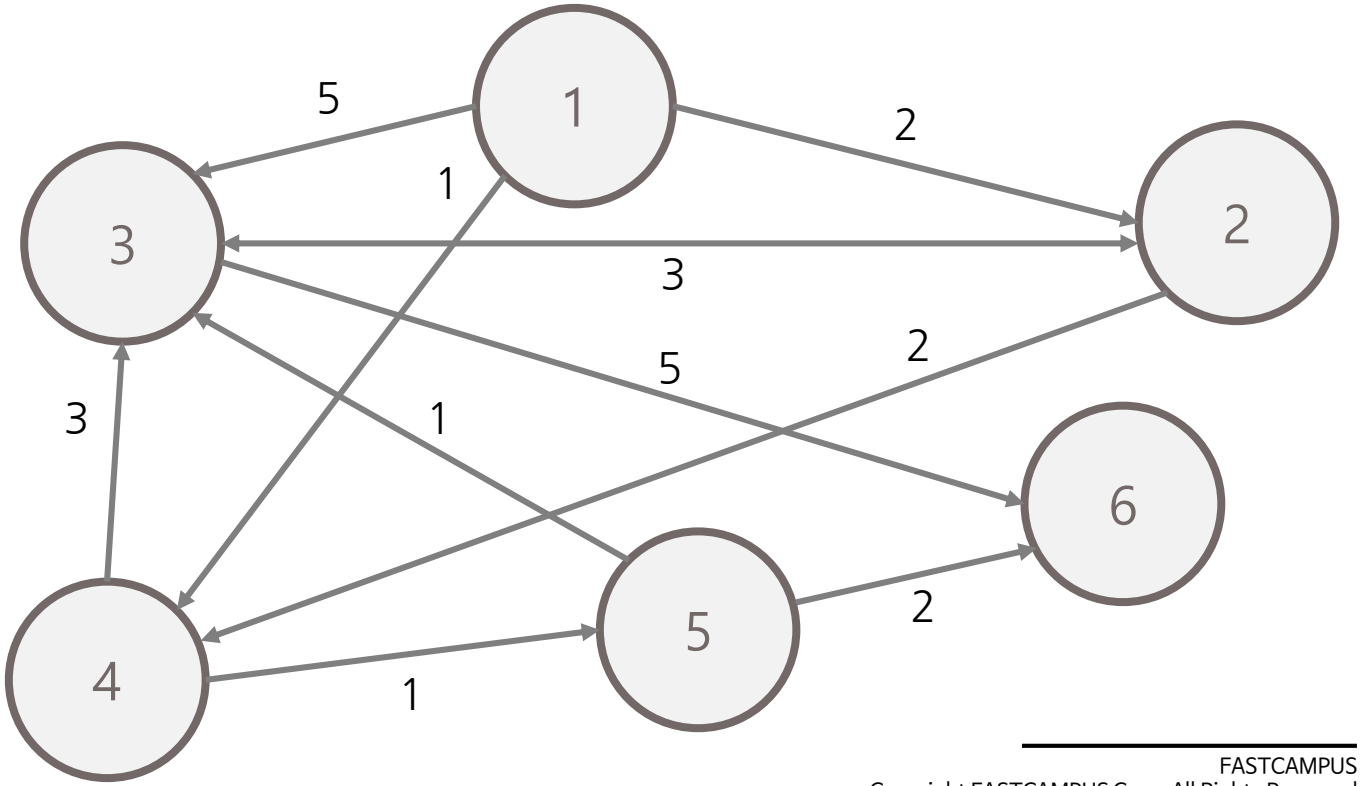
다익스트라의 최단 경로

1) 그래프의 시작점을 선택하여 트리 T에 포함시킵니다.
2) T에 포함된 노드와 T에 포함되지 않은 노드 사이의 간선 중에서 '이동 거리'가 가장 작은 간선을 찾습니다.
3) 해당 간선에 연결된 T에 포함되지 않은 노드를 트리 T에 포함시킵니다.
4) 모든 노드가 포함될 때까지 2)와 3) 과정을 반복합니다.

다익스트라의 최단 경로

다익스트라의 동작 과정

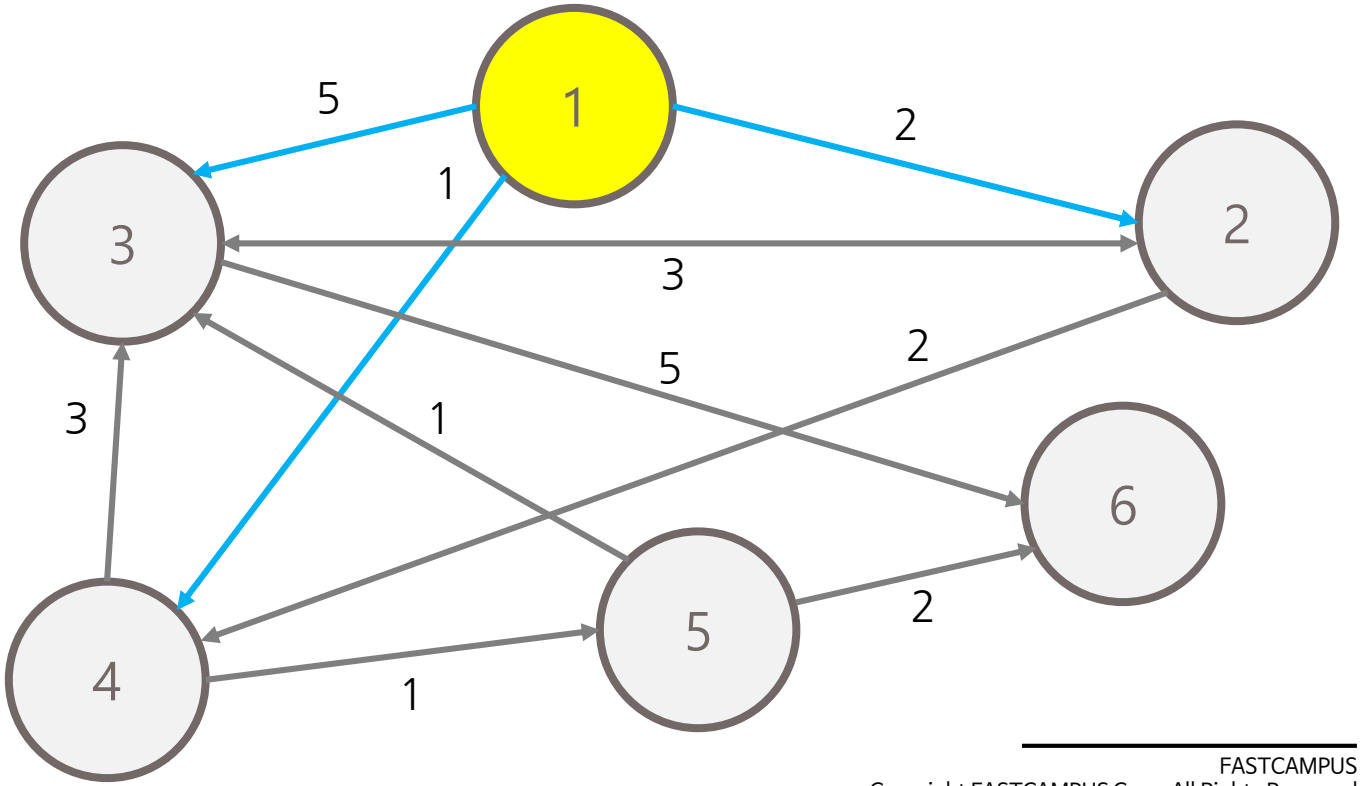
노드	거리
1	
2	
3	
4	
5	
6	



다익스트라의 최단 경로

다익스트라의 동작 과정

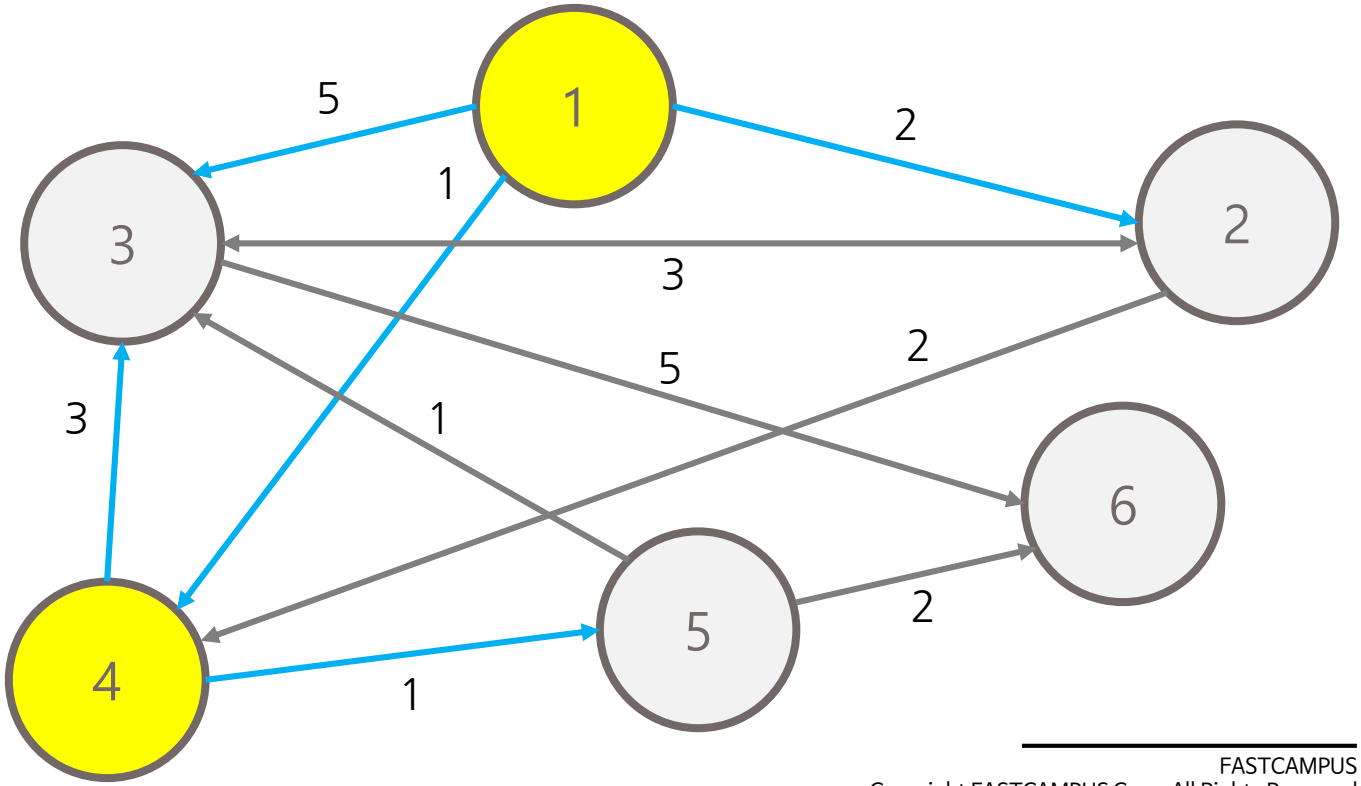
노드	거리
1	0
2	2
3	5
4	1
5	무한
6	무한



다익스트라의 최단 경로

다익스트라의 동작 과정

노드	거리
1	0
2	2
3	4
4	1
5	2
6	무한

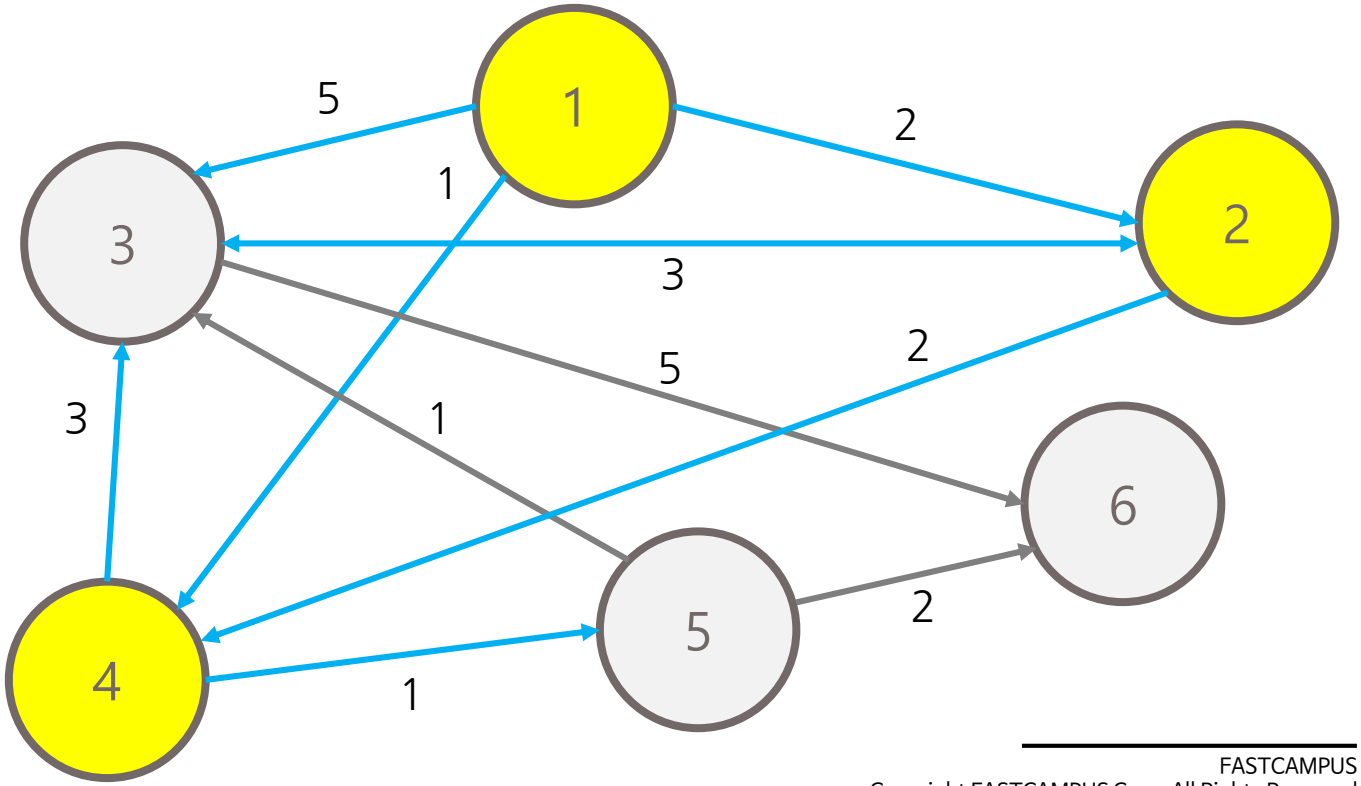


Fast campus

다익스트라의 최단 경로

다익스트라의 동작 과정

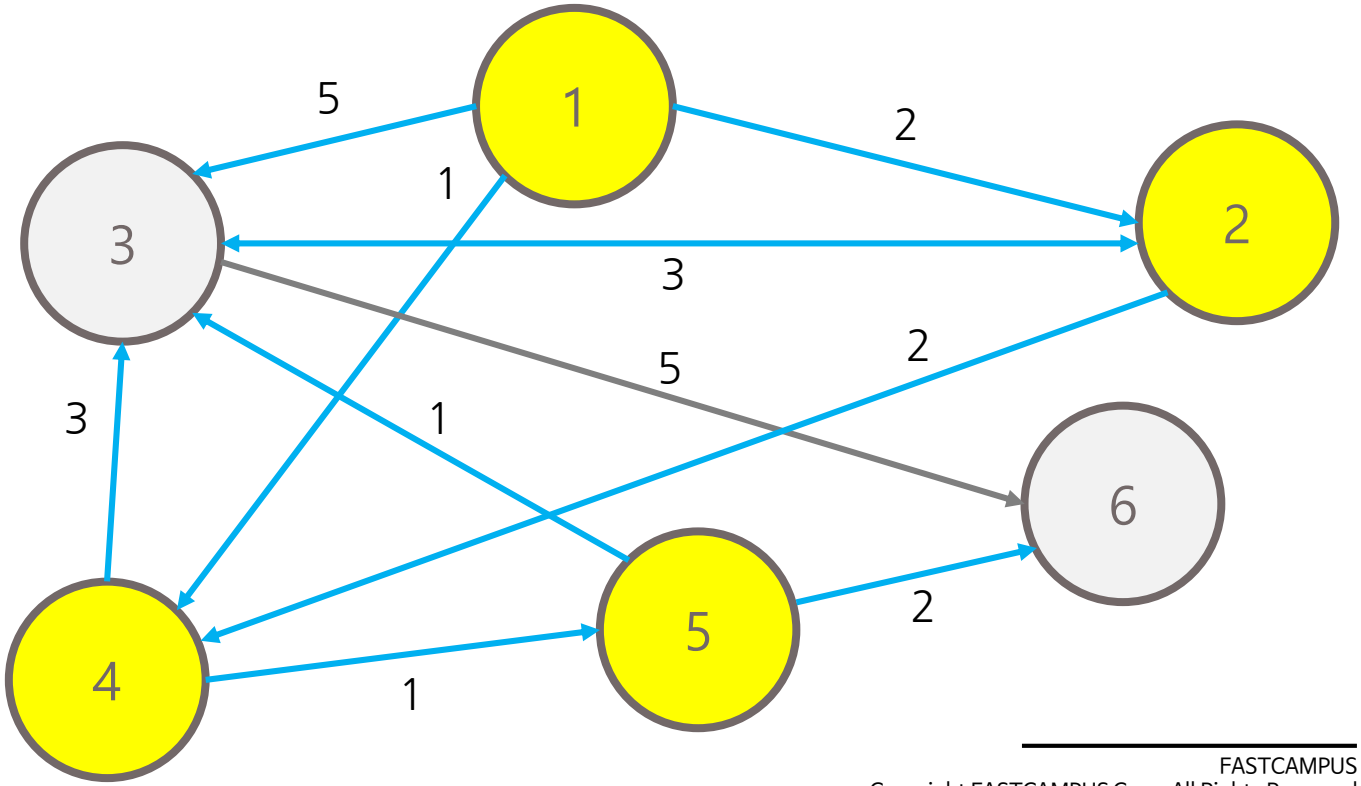
노드	거리
1	0
2	2
3	4
4	1
5	2
6	무한



다익스트라의 최단 경로

다익스트라의 동작 과정

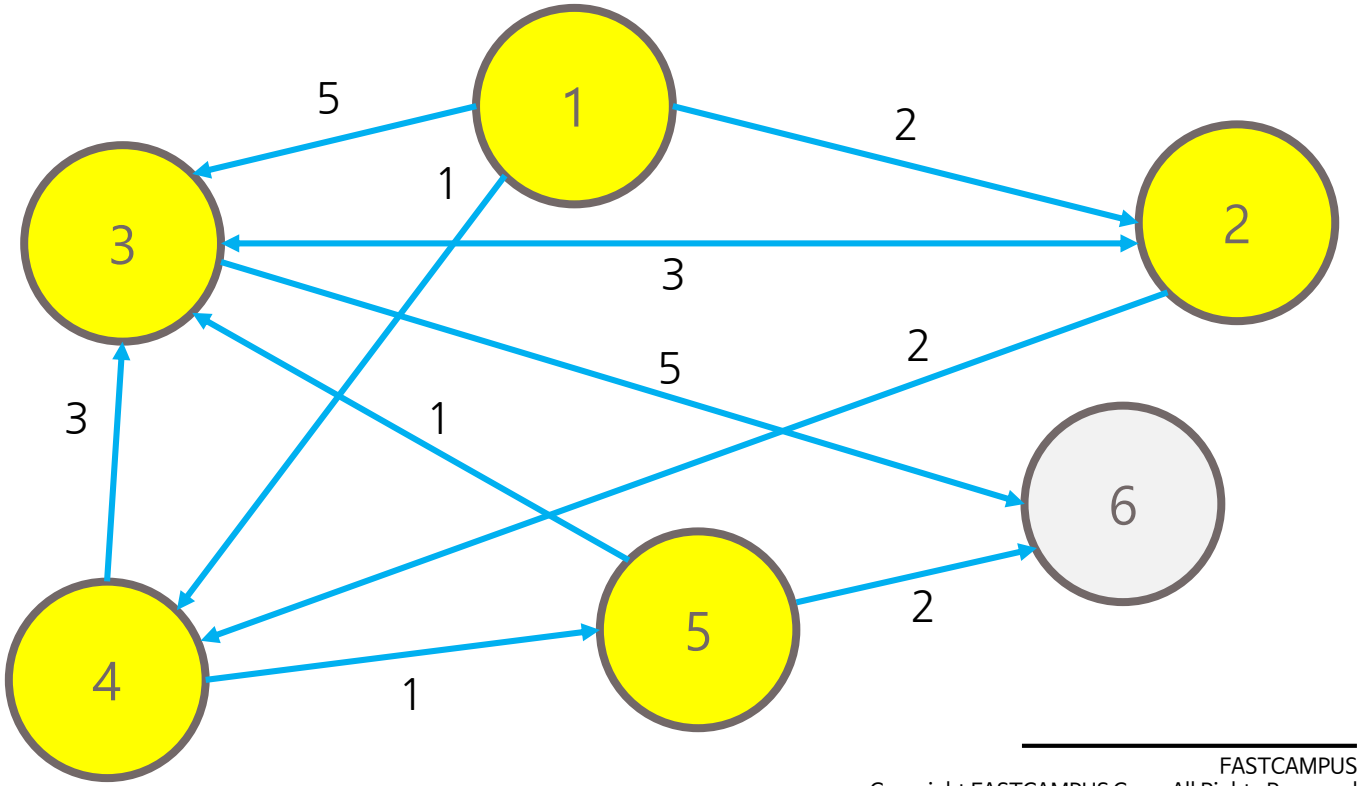
노드	거리
1	0
2	2
3	3
4	1
5	2
6	4



다익스트라의 최단 경로

다익스트라의 동작 과정

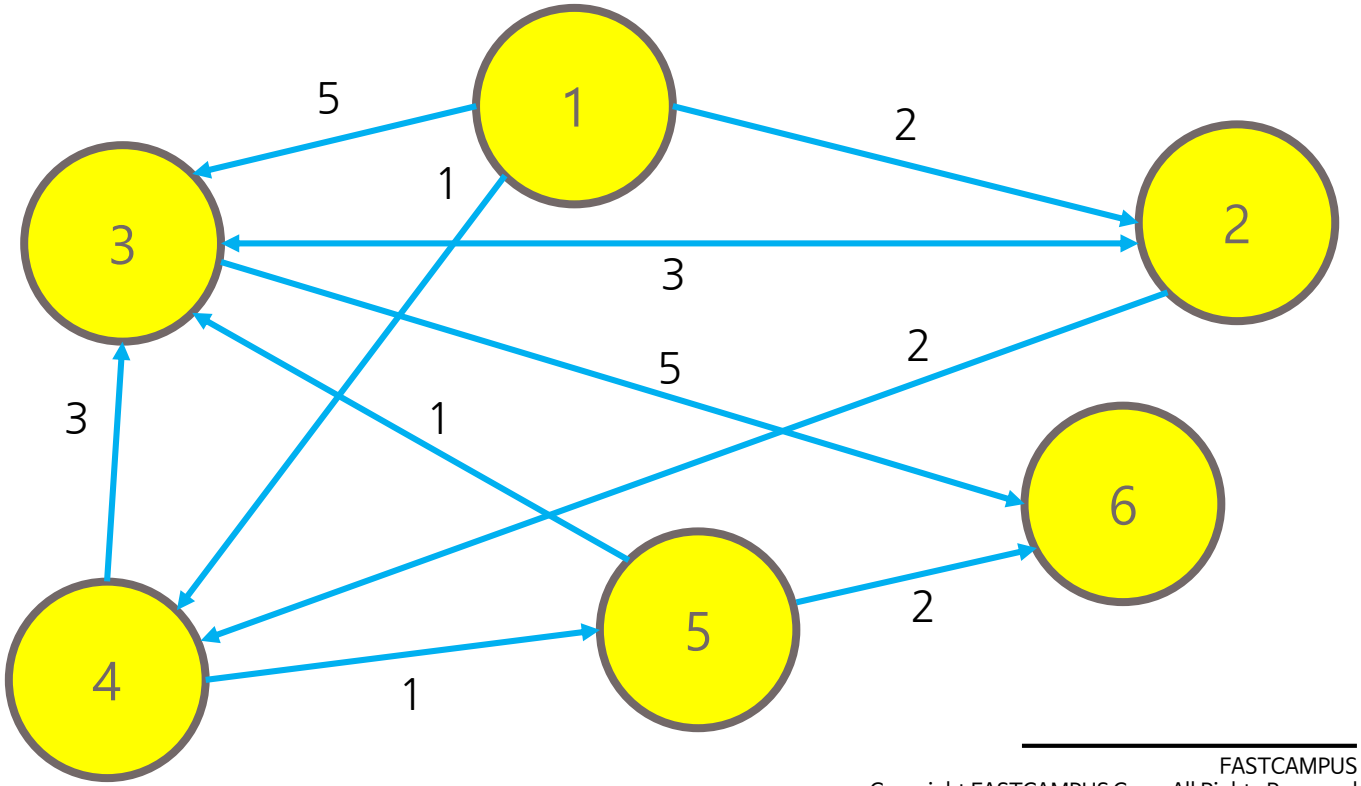
노드	거리
1	0
2	2
3	3
4	1
5	2
6	4



다익스트라의 최단 경로

다익스트라의 동작 과정

노드	거리
1	0
2	2
3	3
4	1
5	2
6	4



다익스트라의 최단 경로

다익스트라의 최단 경로

1) 다익스트라의 알고리즘은 각 간선에 대한 정보를 우선순위 큐에 담아 처리하는 방식으로 구현할 수 있습니다.

다익스트라의 최단 경로

다익스트라 알고리즘 간선 구조체 정의 [최대 노드: 20,000개 / 최대 간선: 300,000개]

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define NODE_MAX 20001
#define EDGE_MAX 600001 // 양방향 간선이므로 300,000개

typedef struct {
  int cost;
  int node;
} Edge;

void swap(Edge *a, Edge *b) {
  Edge temp;
  temp.cost = a->cost;
  temp.node = a->node;
  a->cost = b->cost;
  a->node = b->node;
  b->cost = temp.cost;
  b->node = temp.node;
}
```

다익스트라의 최단 경로

다익스트라 알고리즘 우선순위 큐 정의 및 삽입 함수 구현

```
typedef struct {
  Edge *heap[EDGE_MAX];
  int count;
} priorityQueue;

void push(priorityQueue *pq, Edge *edge) {
  if (pq->count >= EDGE_MAX) return;
  pq->heap[pq->count] = edge;
  int now = pq->count;
  int parent = (pq->count - 1) / 2;
  // 새 원소를 삽입한 이후에 상향식으로 힙을 구성합니다.
  while (now > 0 && pq->heap[now]->cost < pq->heap[parent]->cost) {
    swap(pq->heap[now], pq->heap[parent]);
    now = parent;
    parent = (parent - 1) / 2;
  }
  pq->count++;
}
```

다익스트라의 최단 경로

다익스트라 알고리즘 우선순위 큐 추출 함수 구현

```
Edge* pop(priorityQueue *pq) {
  if (pq->count <= 0) return NULL;
  Edge *res = pq->heap[0];
  pq->count--;
  pq->heap[0] = pq->heap[pq->count];
  int now = 0, leftChild = 1, rightChild = 2;
  int target = now;
  // 새 원소를 추출한 이후에 하향식으로 힙을 구성합니다.
  while (leftChild < pq->count) {
    if (pq->heap[target]->cost > pq->heap[leftChild]->cost) target = leftChild;
    if (pq->heap[target]->cost > pq->heap[rightChild]->cost && rightChild < pq->count) target = rightChild;
    if (target == now) break; // 더 이상 내려가지 않아도 될 때 종료
    else {
      swap(pq->heap[now], pq->heap[target]);
      now = target;
      leftChild = now * 2 + 1;
      rightChild = now * 2 + 2;
    }
  }
  return res;
}
```

다익스트라의 최단 경로

다익스트라 알고리즘 간선 연결 리스트 구현

```
typedef struct Node {
  Edge *data;
  struct Node *next;
} Node;

Node** adj;
int ans[NODE_MAX];

void addNode(Node** target, int index, Edge* edge) {
  if (target[index] == NULL) {
    target[index] = (Node*)malloc(sizeof(Node));
    target[index]->data = edge;
    target[index]->next = NULL;
  }
  else {
    Node* node = (Node*)malloc(sizeof(Node));
    node->data = edge;
    node->next = target[index];
    target[index] = node;
  }
}
```

다익스트라의 최단 경로

다익스트라 알고리즘 사용해보기 ①

```
int main(void) {
  int n, m, k;
  scanf("%d %d %d", &n, &m, &k);
  adj = (Node**)malloc(sizeof(Node*) * (n + 1));
  for (int i = 1; i <= n; i++) {
    adj[i] = NULL;
    ans[i] = INT_MAX;
  }
  priorityQueue *pq;
  pq = (priorityQueue*)malloc(sizeof(priorityQueue));
  pq->count = 0;
  for (int i = 0; i < m; i++) {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    Edge *edge = (Edge*)malloc(sizeof(Edge));
    edge->node = b;
    edge->cost = c;
    addNode(adj, a, edge);
  }
```

다익스트라의 최단 경로

다익스트라 알고리즘 사용해보기 ②

```
  // 다익스트라 알고리즘을 시작합니다.
  ans[k] = 0;
  Edge *start = (Edge*)malloc(sizeof(Edge));
  start->cost = 0; start->node = k; push(pq, start);
  while (1) {
    Edge* now = pop(pq);
    if (now == NULL) break;
    int curNode = now->node;
    int curCost = now->cost;
    if (ans[curNode] < curCost) continue;
    Node* cur = adj[curNode];
    while (cur != NULL) {
      Edge* temp = cur->data;
      temp->cost += curCost;
      if (temp->cost < ans[temp->node]) { ans[temp->node] = temp->cost; push(pq, temp); }
      cur = cur->next;
    }
  }
  for (int i = 1; i <= n; i++) {
    if (ans[i] == INT_MAX) printf("INF\n");
    else printf("%d\n", ans[i]);
  }
  system("pause");
}
```

배운 내용 정리하기

다익스트라의 최단 경로

1) 다익스트라의 최단 알고리즘은 프림 알고리즘과 동일하게 $O(ELogV)$의 시간 복잡도를 가집니다.