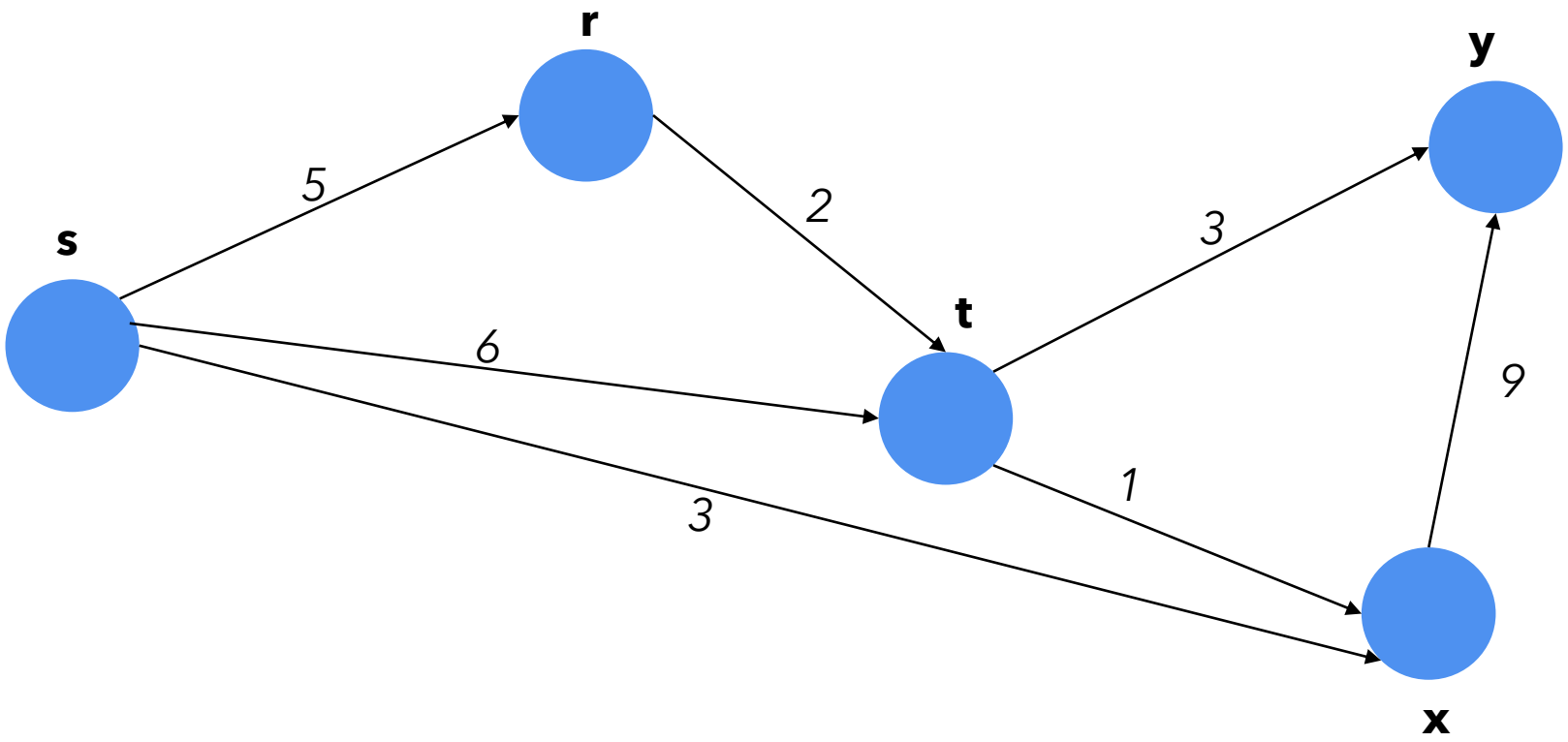
L'algoritmo di Dijkstra

(https://en.wikipedia.org/wiki/Edsger_W._Dijkstra)

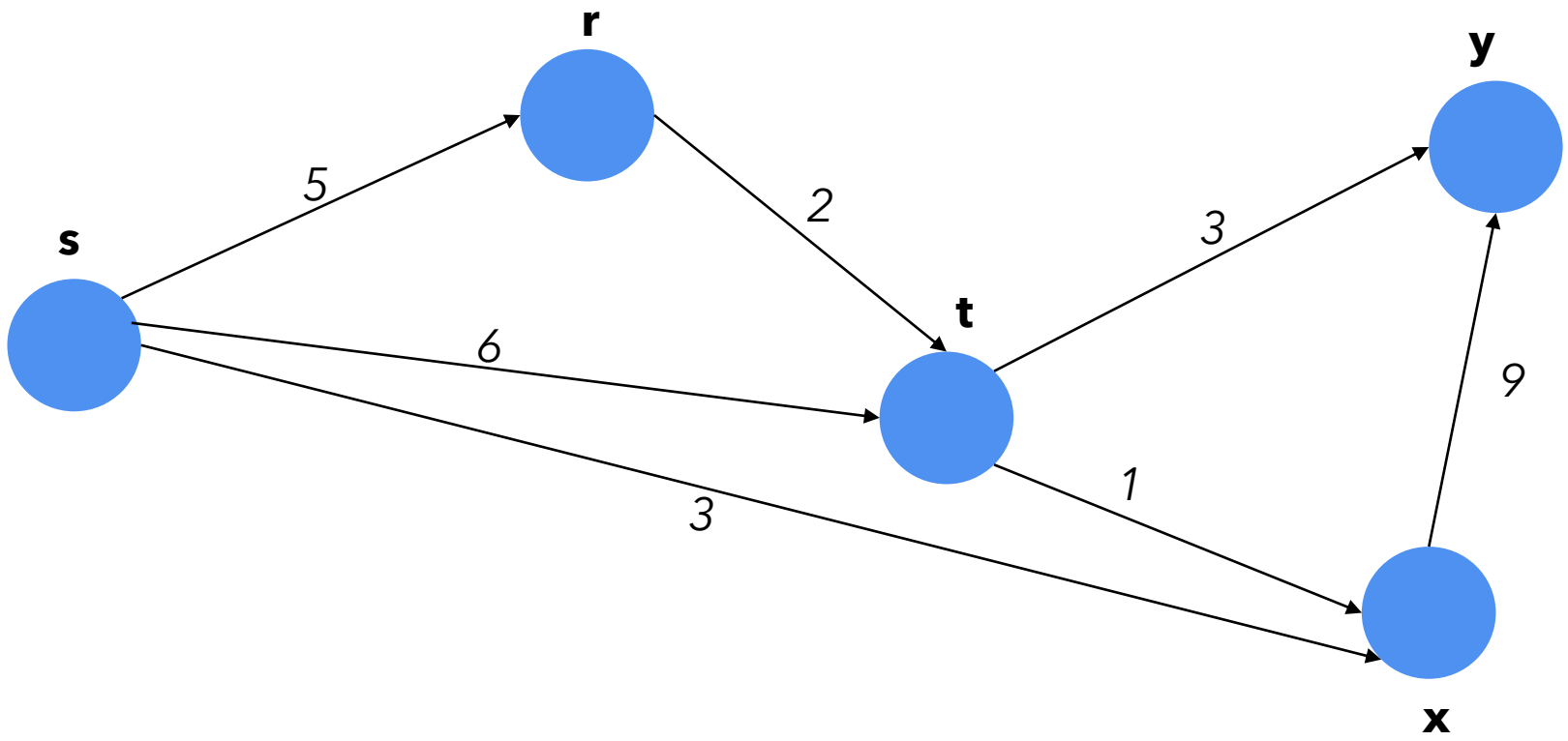
Liceo G.B. Brocchi
Classi seconde Scientifico – opzione scienze applicate
Bassano del Grappa, Maggio 2023
Prof. Giovanni Mazzocchin

Grafi e cammini minimi *(shortest paths)*



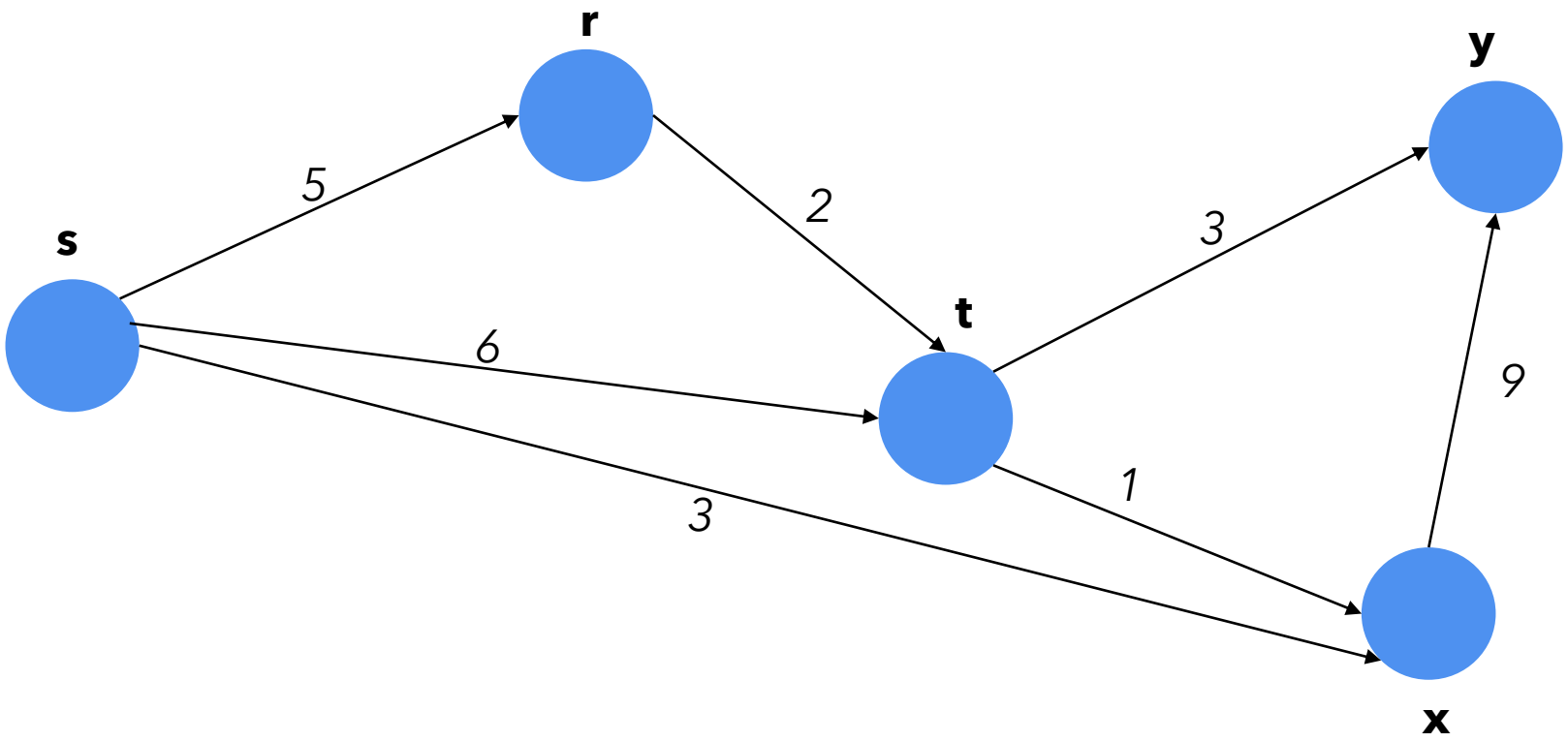
s, **r**, **t**, **x**, **y**: **nodi** (*vertices*) del grafo G
s->r; **r->t**; **t->y**; **s->x**; **t->x**; **s->t; x->y**: archi (*edges*) del grafo G
ad ogni arco è associato un **costo** (numero in corsivo vicino gli archi)

Grafi e cammini minimi *(shortest paths)*



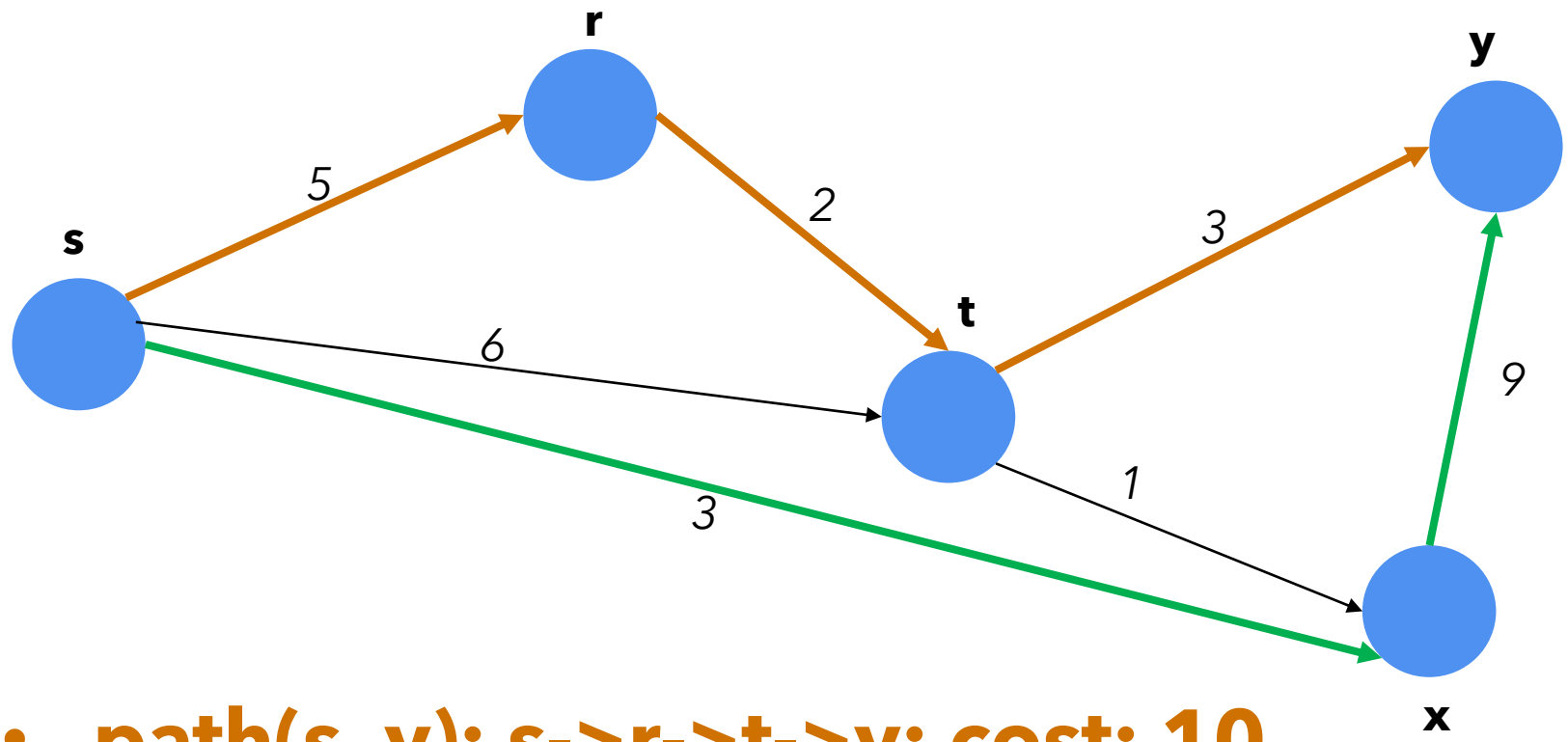
vogliamo trovare il cammino di costo minimo (*shortest path*)
dal nodo **s** al nodo **y**

Grafi e cammini minimi *(shortest paths)*



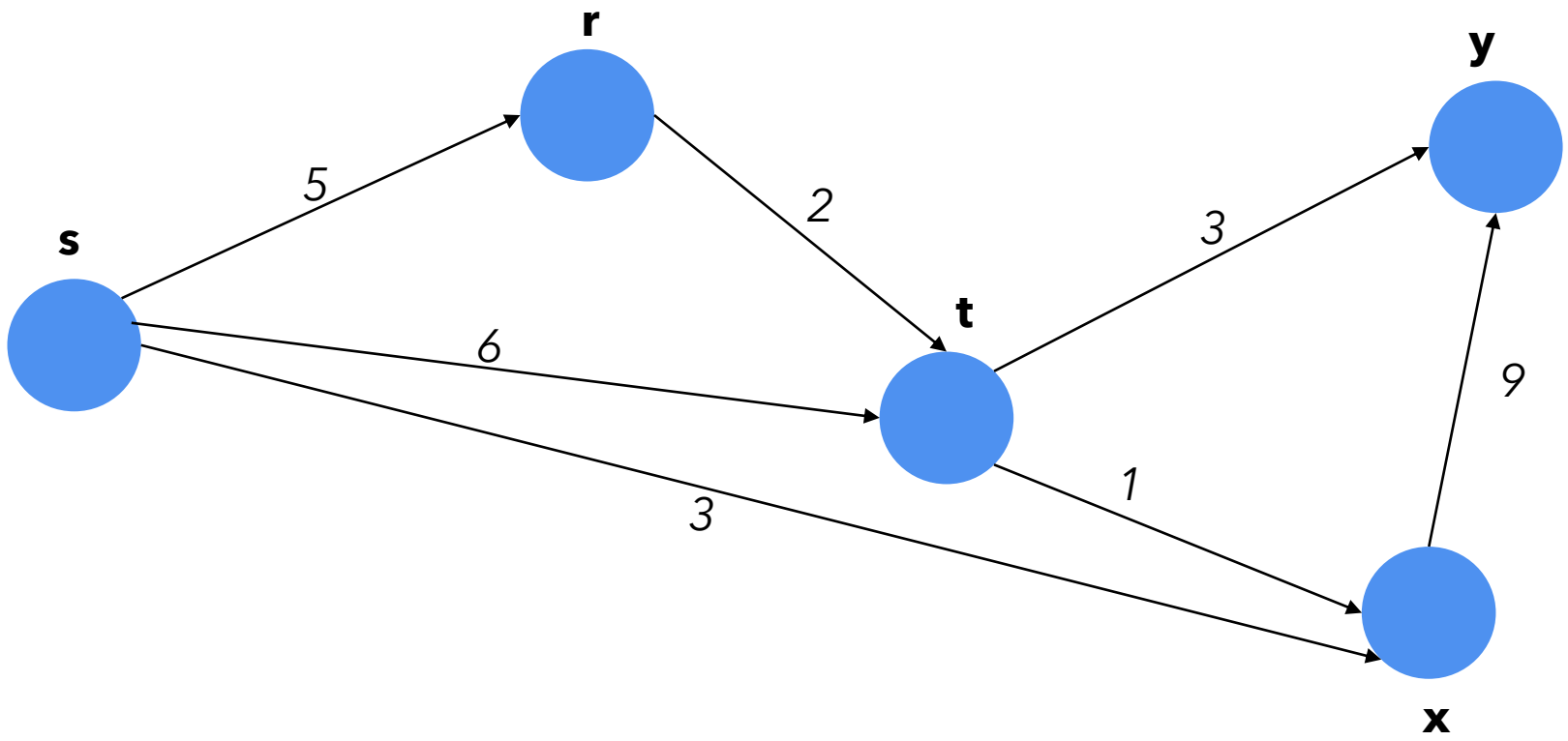
pensate ad alcune applicazioni informatiche (o nel campo delle telecomunicazioni) che potrebbero aver bisogno di trovare cammini minimi…

Grafi e cammini minimi *(shortest paths)*



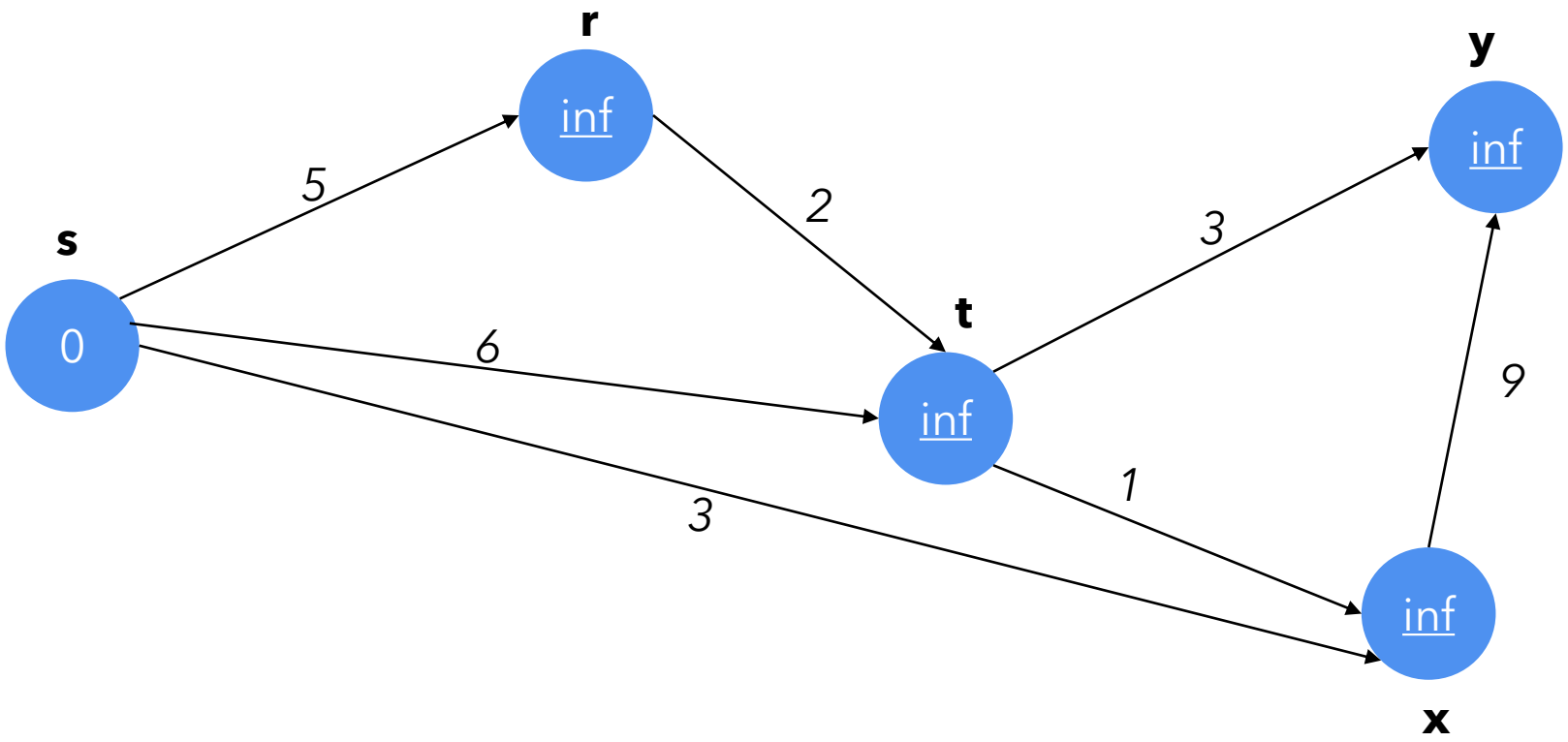
- **path(s, y): s->r->t->y; cost: 10**
- **path(s, y): s->x->y; cost: 12**
- **etc...**

Grafi e cammini minimi *(shortest paths)*



all'interno di un nodo **v** scriveremo il costo del percorso **s->v** di costo minimo noto finora

Grafi e cammini minimi *(shortest paths)*



prima di applicare un qualunque algoritmo, assumiamo che il costo di **s->v** (con v != s) sia +infinito; ovviamente il costo del percorso minimo **s->s** è 0 e non cambierà mai

L'algoritmo di Dijkstra – descrizione in lingua naturale

Initialization steps:

1. Consider a directed, weighted graph G = (V, E), where V is the set of vertices, E is the set of edges;
2. Select a source node s. Shortest paths will be computed with s as source;
3. Each v of V will contain the minimum cost of s->v computed so far (+inf at the beginning);
4. Create a set called Q and put Q = V (i.e. assign the set of vertices of G to Q);

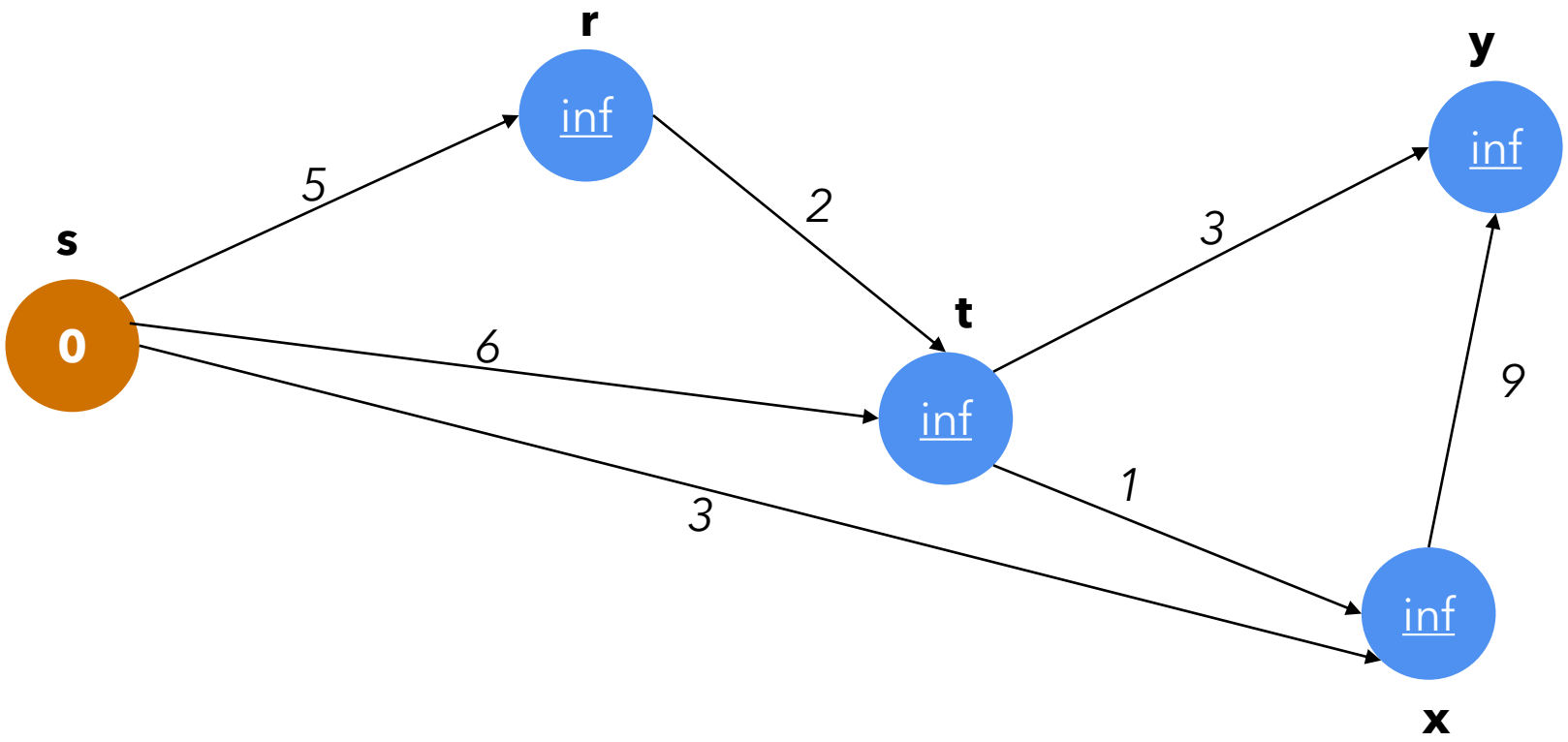
5. **NB: costs must be all >= 0**

L’algoritmo di Dijkstra – descrizione in lingua naturale

Procedure:

```
As long as Q is not empty:
   1. extract the node of Q with minimum cost, call it v
   2. for each u, successor of v:
          if v.cost + cost(v, u) < u.cost:
              set u.cost = v.cost + cost(v, u)
              update Q accordingly
```

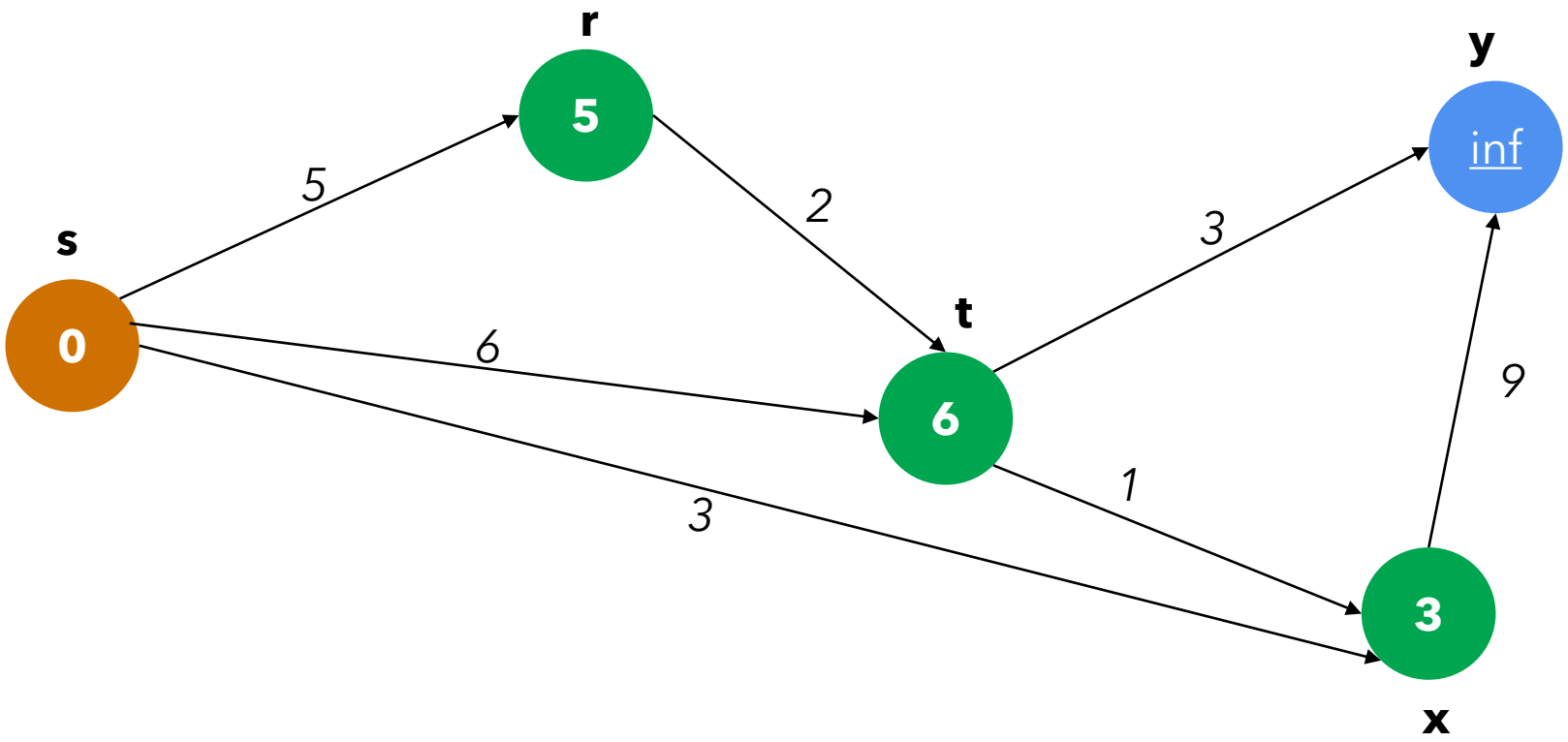
L'algoritmo di Dijkstra - esecuzione



vertice da cui parte la visita ai successori

Q: (s,0), (r,inf), (t,inf), (x,inf), (y,inf)

L'algoritmo di Dijkstra - esecuzione

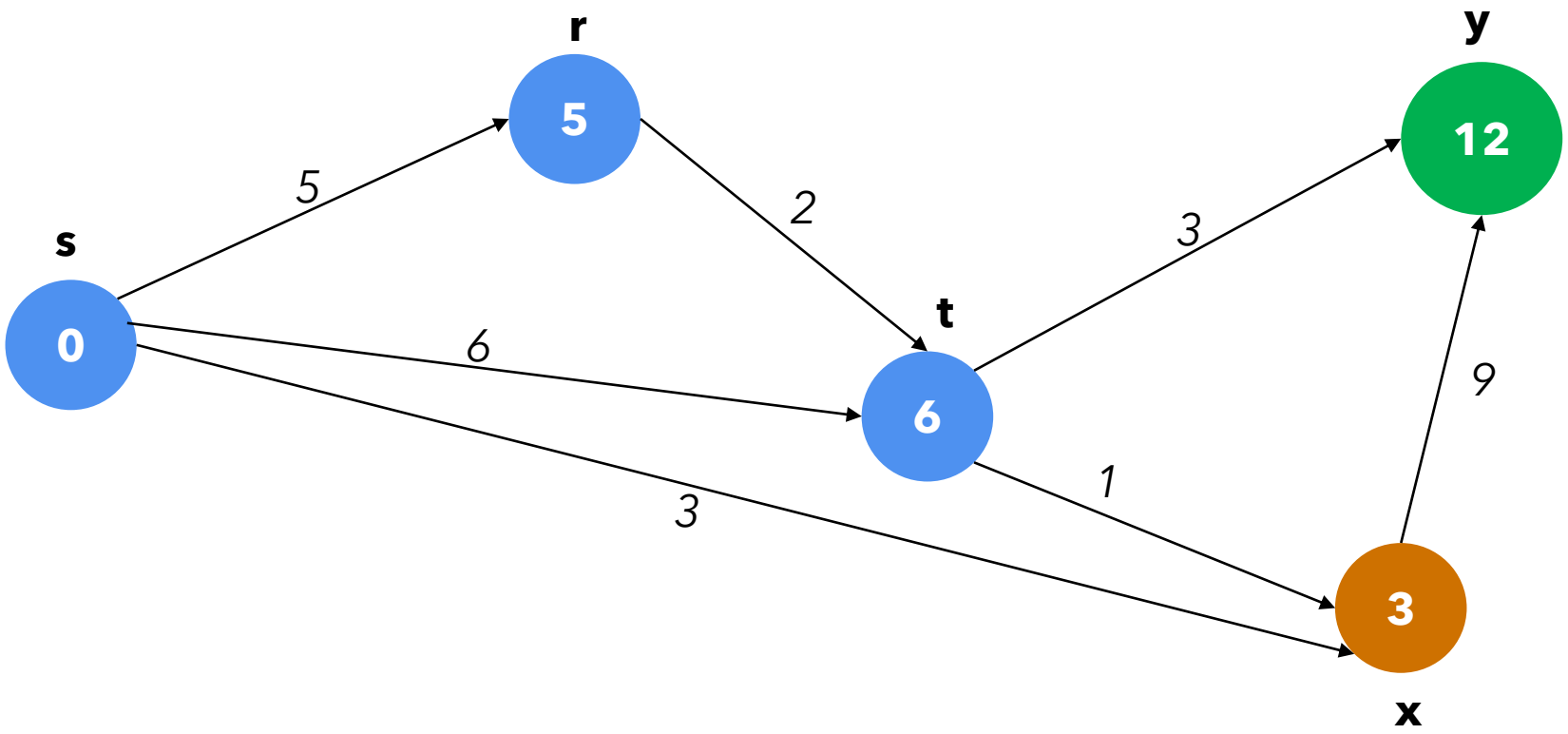


vertice da cui parte la visita ai successori

vertici visitati ed eventualmente aggiornati

```
Q: (r,5), (t,6), (x,3), (y,inf)
pred(r): s; pred(t): s; pred(x): s
```

L'algoritmo di Dijkstra - esecuzione

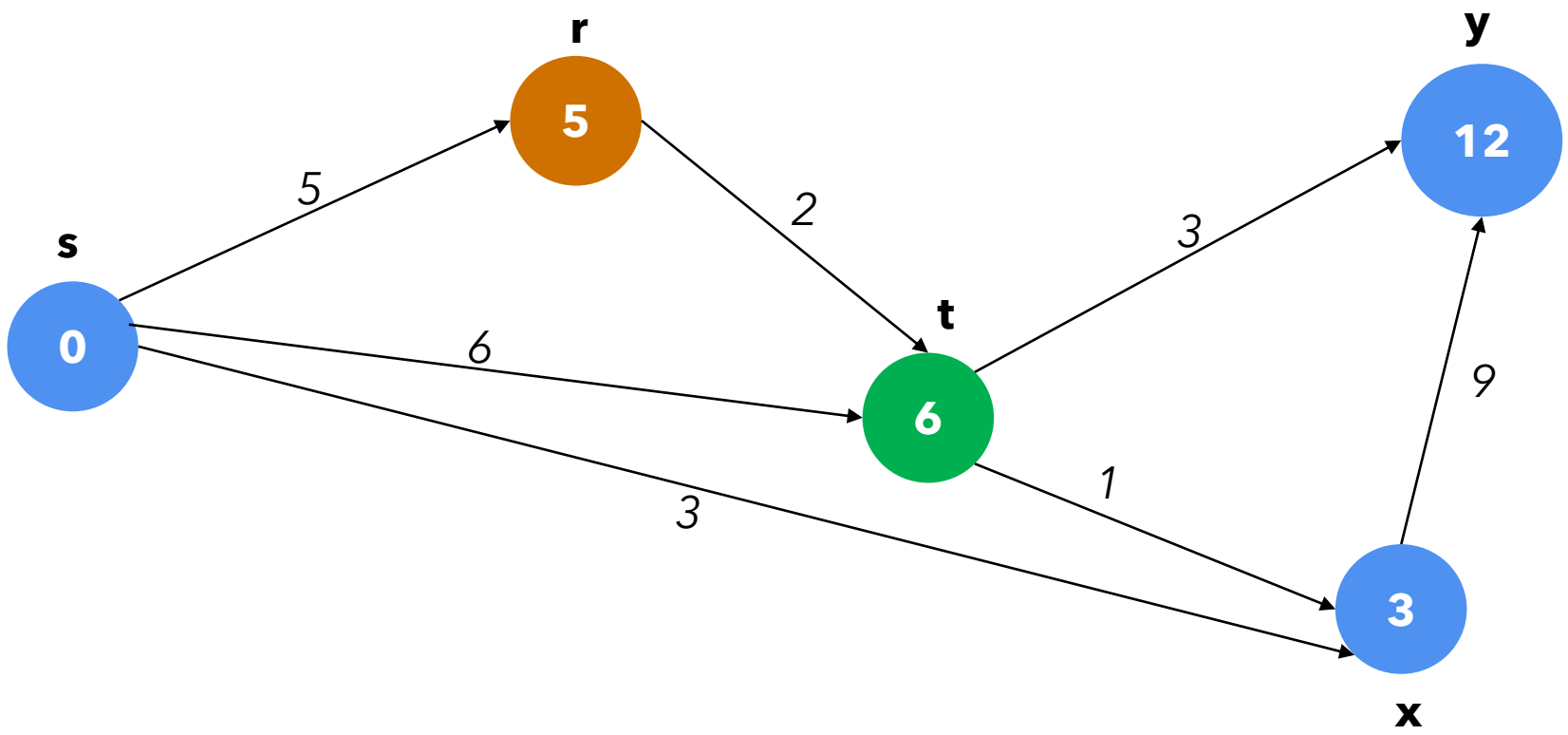


vertice da cui parte la visita ai successori

vertici visitati ed eventualmente aggiornati

```
Q: (r,5), (t,6), (y,12)
pred(y): x
```

L'algoritmo di Dijkstra - esecuzione

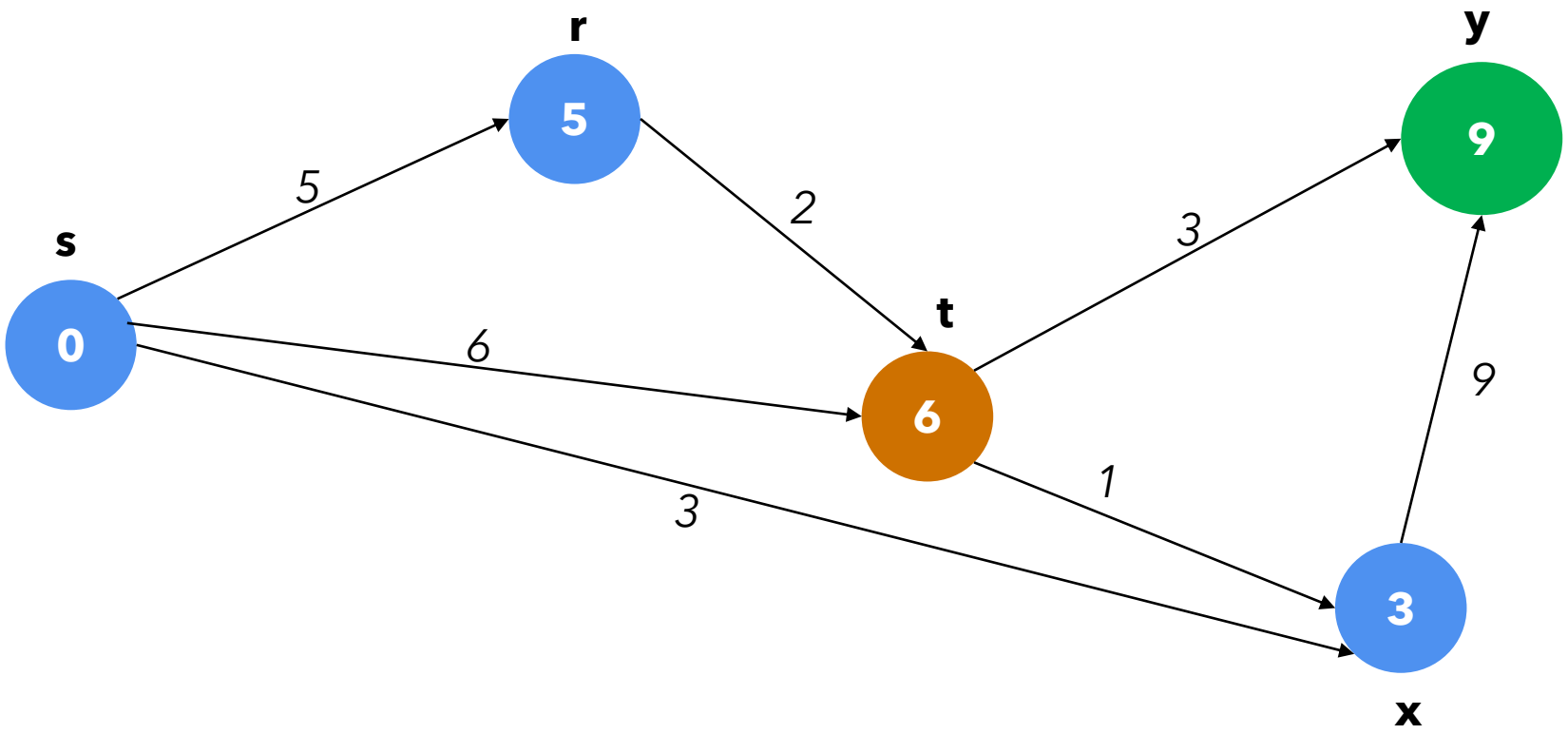


vertice da cui parte la visita ai successori

vertici visitati ed eventualmente aggiornati

Q: (t,6), (y,12)

L'algoritmo di Dijkstra - esecuzione



vertice da cui parte la visita ai successori

vertici visitati ed eventualmente aggiornati

```
Q: (y,9)
pred(y): t
```

Scriviamolo in Python!

- Rappresentiamo il grafo con un dizionario
- Vanno memorizzate anche le informazioni relative ai costi minimi dalla sorgente scelta, per ogni vertice
- Cerchiamo di scrivere un codice leggibile e simile allo pseudocodice
- Non è facile, ma con qualche dritta si può fare

- https://github.com/Cyofanni/high-school-cs-class/blob/main/python/optimization/dijkstra.py