- 书面作业讲解
  - TC第25.1节练习4、5、6、9、10
  - TC第25.2节练习2、4、6、8
  - TC第25.3节练习2、3
  - TC第25章问题2

TC第25.1节练习4

25.1-4

Show that matrix multiplication defined by EXTEND-SHORTEST-PATHS is associative.

```
EXTEND-SHORTEST-PATHS(L, W)
1  n = L.rows
2  let L' = (l'_ij) be a new n × n matrix
3  for i = 1 to n
4      for j = 1 to n
5          l'_ij = ∞
6          for k = 1 to n
7              l'_ij = min(l'_ij, l_ik + w_kj)
8  return L'
```

$$
\begin{aligned}
l^{(m-1)} &\rightarrow a\,, \\
w &\rightarrow b\,, \\
l^{(m)} &\rightarrow c\,, \\
\min &\rightarrow +\,, \\
+ &\rightarrow \cdot
\end{aligned}
$$

TC第25.1节练习5

25.1-5

Show how to express the single-source shortest-paths problem as a product of matrices and a vector. Describe how evaluating this product corresponds to a Bellman-Ford-like algorithm (see Section 24.1).

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
```

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2      v.d = u.d + w(u, v)
3      v.π = u
```

```
EXTEND-SHORTEST-PATHS(L, W)
1  n = L.rows
2  let L' = (l'_ij) be a new n × n matrix
3  for i = 1 to n
4      for j = 1 to n
5          l'_ij = ∞
6          for k = 1 to n
7              l'_ij = min(l'_ij, l_ik + w_kj)
8  return L'
```

TC第25.1节练习6

25.1-6

Suppose we also wish to compute the vertices on shortest paths in the algorithms of this section. Show how to compute the predecessor matrix Π from the completed matrix L of shortest-path weights in $O(n^3)$ time.

for i…

for j…

for k…

if (L_{ij}==L_{ik}+W_{kj})

then Π_{ij}=k

可不可以改成

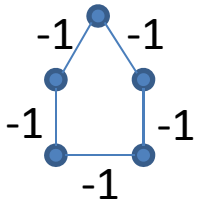
if (L_{ij}==L_{ik}+L_{kj})

then Π_{ij}=Π_{kj}

TC第25.1节练习9

25.1-9
Modify FASTER-ALL-PAIRS-SHORTEST-PATHS so that it can determine whether the graph contains a negative-weight cycle.

- 检查结果对角线上是否有负值，行不行？
  - n=5, m=4时，运行结束



FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
1 $n = W.rows$
2 $L^{(1)} = W$
3 $m = 1$
4 **while** $m < n - 1$
5 let $L^{(2m)}$ be a new $n \times n$ matrix
6 $L^{(2m)} =$ EXTEND-SHORTEST-PATHS$(L^{(m)}, L^{(m)})$
7 $m = 2m$
8 **return** $L^{(m)}$

TC第25章问题2

25-2 Shortest paths in ϵ-dense graphs

A graph $G = (V, E)$ is *ϵ-dense* if $|E| = \Theta(V^{1+\epsilon})$ for some constant ϵ in the range $0 < \epsilon \leq 1$. By using d-ary min-heaps (see Problem 6-2) in shortest-paths algorithms on ϵ-dense graphs, we can match the running times of Fibonacci-heap-based algorithms without using as complicated a data structure.

a. What are the asymptotic running times for INSERT, EXTRACT-MIN, and DECREASE-KEY, as a function of d and the number n of elements in a d-ary min-heap? What are these running times if we choose $d = \Theta(n^{\alpha})$ for some constant $0 < \alpha \leq 1$? Compare these running times to the amortized costs of these operations for a Fibonacci heap.

- INSERT: 代价=树高=$\log_d n = 1/\alpha$
- DECREASE-KEY: 代价=INSERT=$\log_d n = 1/\alpha$
- EXTRACT-MIN: 代价=分支因子*树高=$d\log_d n = n^{\alpha}/\alpha$

TC第25章问题2 (续)

b. Show how to compute shortest paths from a single source on an ϵ-dense directed graph $G = (V, E)$ with no negative-weight edges in $O(E)$ time. (*Hint:* Pick d as a function of ϵ.)

- V次EXTRACT-MIN + E次DECREASE-KEY
- 取d=V^{ε}： $O(V*V^{\varepsilon}/\varepsilon + E*1/\varepsilon) = O(V^{1+\varepsilon}/\varepsilon+V^{1+\varepsilon}/\varepsilon) = O(V^{1+\varepsilon}) = O(E)$

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

TC第25章问题2 (续)

c. Show how to solve the all-pairs shortest-paths problem on an ϵ-dense directed graph $G = (V, E)$ with no negative-weight edges in $O(VE)$ time.

- (b)运行V次：V*O(E) = O(VE)

TC第25章问题2 (续)

d. Show how to solve the all-pairs shortest-paths problem in $O(VE)$ time on an ϵ-dense directed graph $G = (V, E)$ that may have negative-weight edges but has no negative-weight cycles.

- Johnson算法： Bellman-Ford + V次Dijkstra
- O(VE) + O(VE) = O(VE)

- 教材讨论
  - DW第4章

问题1：connectivity

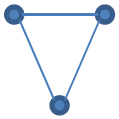
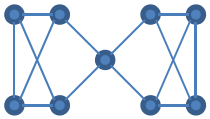
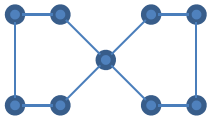
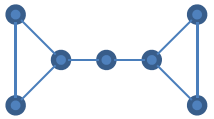
- connectivity=k是什么意思？
- 完全图的connectivity是多少？
- 你能分别举出connectivity为0、1、2、3的非完全图吗？
- k-connected又是什么意思？

问题1：connectivity (续)

- edge-connectivity=k是什么意思？
- 完全图的edge-connectivity是多少？
- 你能分别举出edge-connectivity为0、1、2、3的非完全图吗？
- k-edge-connected又是什么意思？

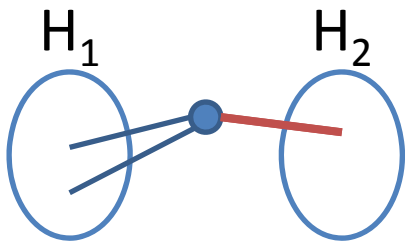
问题1： connectivity (续)

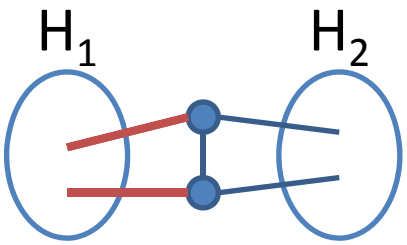
- $\kappa \le \kappa' \le \delta$，你能分别举出例子吗？
  - $\kappa = \kappa' = \delta$
  - $\kappa < \kappa' < \delta$
  - $\kappa < \kappa' = \delta$
  - $\kappa = \kappa' < \delta$

问题1： connectivity (续)

- 结合这两个图，你能说明为什么3-正则图满足κ=κ’吗？

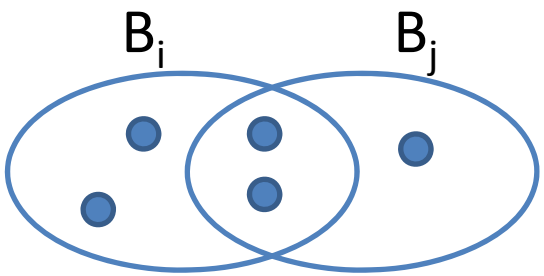


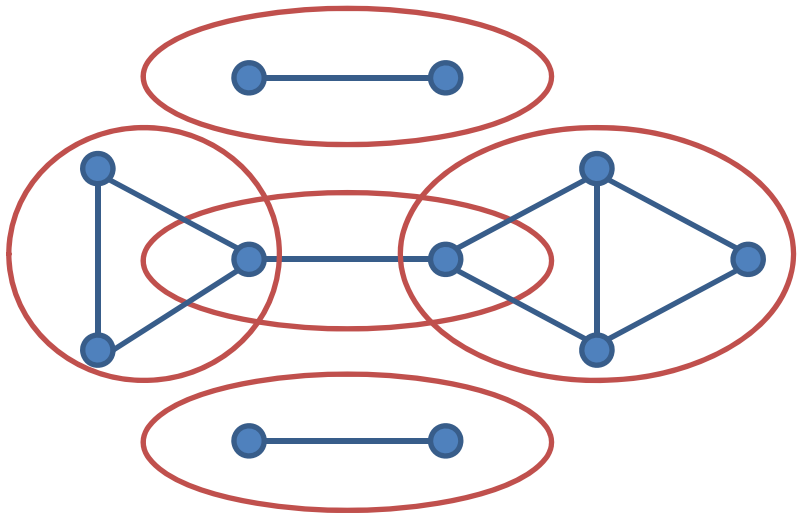


问题2：block

- 什么叫block？
- block = maximal 2-connected subgraph?
- 为什么两个block最多只有一个公共顶点？

- 我们可以构建一个“block-cut-vertex graph”，其中：
  - 顶点是block和cut vertex
  - block和cut vertex之间有边当且仅当这个cut vertex在这个block中
- 你认为这个graph有什么特征？





问题2： block (续)

- Algorithm for computing blocks (biconnected components)
- 你读懂了吗？

The idea is to run a depth-first search while maintaining the following information:

1. the depth of each vertex in the depth-first-search tree (once it gets visited), and
2. for each vertex v, the lowest depth of neighbors of all descendants of v in the depth-first-search tree, called the **lowpoint**.

The depth is standard to maintain during a depth-first search. The lowpoint of v can be computed after visiting all descendants of v (i.e., just before v gets popped off the depth-first-search stack) as the minimum of the depth of v, the depth of all neighbors of v (other than the parent of v in the depth-first-search tree) and the lowpoint of all children of v in the depth-first-search tree.

The key fact is that a nonroot vertex v is a cut vertex (or articulation point) separating two biconnected components if and only if there is a child y of v such that $\text{lowpoint}(y) \geqslant \text{depth}(v)$. This property can be tested once the depth-first search returned from every child of v (i.e., just before v gets popped off the depth-first-search stack), and if true, v separates the graph into different biconnected components. This can be represented by computing one biconnected component out of every such y (a component which contains y will contain the subtree of y, plus v), and then erasing the subtree of y from the tree.

The root vertex must be handled separately: it is a cut vertex if and only if it has at least two children. Thus, it suffices to simply build one component out of each child subtree of the root (including the root).

问题3：k-connected graphs

- 什么叫x,y-cut？
- 你能复述Menger定理的内容吗？
- 根据Menger定理，你认为图的connectivity和图中顶点间的不交路之间存在什么联系？