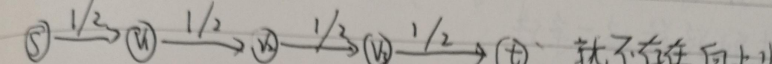
Exercise 1

1.1.

不是每个流网络都有一个向上临界边. 如



此不存在向上临界边.

算法:

初始化 f 为0流, 即 $f(u,v)=0$ 对 $\forall (u,v)\in E$ 成立; //$O(|E|)$

$G_f \leftarrow G$;

While G_f 中存在 s-t 路径到 p Do // 循环最多进行 $|f|$ 次.

$\varepsilon \leftarrow$ 增广路径 p 的公差

For p 上每条边 (u,v) Do // $O(|E|)$

If $(u,v)\in E$ Then

$f(u,v) \leftarrow f(u,v)+\varepsilon$;

对 G_f 进行相应的修改

Else

$f(v,u) \leftarrow f(v,u)-\varepsilon$;

对 G_f 进行相应的修改

令 $S=$ 从 s 在 G_f 中存在 s 到 u 的路径 $\}$. //$O(|V||E|)$

对对 $\forall v\in V-S$ 在 G_f 中存在 v 到 t 的路径. //$O(|V||E|)$

则 $\forall v_i\in S$, $v_j\in V-S$

若 $\exists v_iv_j\in E$, 则 v_iv_j 为向上临界边

算法复杂性分析, 由上述注释可知 $T_n=O(|E|(|V|+|f|))$

正确性证明: 前面部分使用 Ford-Fulkerson 算法得到 $S, V-S$ 为最小割. 而算法中 $\forall v\in V-S$, G_f 中存在 v 到 t 的路径保证了其它路上没有剩余容量, 故增大最小割的容量即可增大最大流的流量. 故此时, $\exists v_iv_j\ (v_i\in S, v_j\in V-S, v_iv_j\in E)$ 满足向上临界边的定义.

1.2

方法不完全相同. 对于向下临界边，只须要其为最小割 $[S, V-S]$ 中的 v_iv_j ($v_i \in S$, $v_j \in V-S$, 且 $v_iv_j \in E$) 即可，不需要 $\forall v \in V-S$, G_f 中存在 V 到 t 的路径这一条件.

算法:

同 1.1 由 Ford-Fulkerson 算法得到最小割 $[S, V-S]$

若 $\exists v_i \in S$, $v_j \in V-S$, $v_iv_j \in E$.

则 v_iv_j 为向下临界边.

时间复杂性. $T_n = O(|f||E|)$.

正确性证明: 由 Ford-Fulkerson 算法得到的必为最小割 $[S, V-S]$.

且由最大流最小割定理，最小割 $[S, V-S]$ 的容量为 G 的最大流流量. 故对 v_iv_j ($v_i \in S$, $v_j \in V-S$, $v_iv_j \in E$) 降低其容量，最大流流量必然下降. 故 v_iv_j 满足向下临界边的定义.

Exercise 2.

由最大流最小割定理可知，若 G中存在唯一最大流，则G中可存在唯一最小割.

算法：

初始化f为0流，即 $f(u,v)=0$ 对 $\forall (u,v)\in E$ 成立； //O(E)

$G_f \leftarrow G$;

While G_f中存在 s-t路径到 P Do //循环次数最多为|f|

$\varepsilon \leftarrow$ 增广路径P的公差

For P上的每条边 (u,v) Do

If $(u,v)\in E$ Then //O(E)

$f(u,v) \leftarrow f(u,v)+\varepsilon$;

对G_f进行相应的修改;

Else

$f(v,u) \leftarrow f(v,u)-\varepsilon$;

对G_f进行相应的修改;

令 S={u|在G_f中存在从s到u的路径} O(|V|·|E|)

If $V-S \neq \emptyset$

存在 f为最大流，[S, V-S]为最小割].

初始化步骤时间复杂度为O(|E|)，循环时间复杂度为 O(|f|·|E|)

未步时间复杂度为O(|V|·|E|) 故总时间复杂度：

$T_n = O(|f|\cdot|E|)$

Exercise 3.

Push - Relable 算法：

引理 1：设 $G(V, E)$ 为一个源结点为 s 汇点为 t 的流网络，f 为一个预流，设 h 为 V 上的高度函数，那么剩余网络 G_f 中不存在一条从源结点 s 到汇点 t 的路径。（课件上已证明）

定理：若 Push - Relable 算法在图上运行时能够终止，则算法计算出的预流 f 是图 G 的一个最大流。

证明：定义循环不变量：每次算法执行 while 循环时，f 都是 G 的一个预流。

初始化：f 是图 G 的一个预流

保持：循环体中唯一的操作是 push 和 relable

(1) relable 操作只影响结点高度，不影响流值，因此不影响 f 是否是一个预流。

(2) 由 push 定义可知，若调用前 f 是一个预流，则调用后 f 仍然是一个预流。

因此 循环不变量得到维持

终止：算法终止时，$V - \{s, t\}$ 中每个结点的超额流必定为 0，图中不存在溢出结点，f 是图 G 的一个流

再根据前面引理，剩余网络中不存在一条从 s 到 t 的路径

由最大流 - 最小割定理知，f 是图 G 的最大流。

Relable - to - front 算法：

定理：Relable - to - front 算法终止后，最后的预流是最大流。

证明：初始化：算法初始化是预流。

循环：每次推送或重标操作后，仍是预流。

终止：算法结束后得到一个预流 f

算法结束后 $e(u)=0$ 对任意 $u \neq s, t$ 成立，故剩余网络中不存在 s-t 路径，由最大流-最小割定理，f 是最大流。

Exercise 4.

定理：Hopcroft-Karp 算法算法在图上运行时能够终止，且最后得到的 M 是最大二分匹配

初始化：算法初始化 M 是一个匹配

循环：算法中每次循环搜索极多的无公共顶点最短增广路，全部替换得到的更大的匹配 M，M 仍为匹配

终止：当无法得到更大的匹配时，算法终止，且由算法过程确定最短增广路长度与潜在终点知，M 为最大匹配

时间复杂度：寻找潜在顶点 $O(|V|^2)$，对增广路进行增广 $O(|E|)$

故总时间复杂度 $T_n = O(|V|^2|E|)$