Exercise 1.2

Consider a graph with n vertices where the vertices can be partitioned into two subsets, S and T, of size $n/2$ (we assume for simplicity that n is even), such that the subgraphs consisting of these vertices and the edges between them are both complete.
Furthermore there are two vertices, one in each subgraph, call them s and t resp., that are connected by an edge, and no other vertex is connected to any other vertex of the other subgraph.
We call such a graph a *handweight graph*.

Clearly a handweight graph has a unique min-cut of size 1, performed by cutting the edge connecting s and t.
Furthermore, if a vertex from S is merged with a vertex from T the min-cut size is increased.

Exercise 1.3

Let V be the verifier algorithm outputting 0 if the solution is correct and 1 otherwise. Then the algorithm runs on an instance, I, in the following steps:

$v \leftarrow 0$
while $v = 0$
    $s \leftarrow A(I)$
    $v \leftarrow V(s)$
return s

This procedure clearly returns a correct solution if it terminates, though, a priori, it is not guaranteed that it terminates.

Note that at each iteration of the while loop A returns a correct solution with probability γ, hence the probability that the verifyer returns 1 is γ.
Since the algorithm exists the while loop the first time a correct solution is found, i.e. the first time the verifier returns 1, the number of iterations in the while loop is clearly geometrically distributed with success probability γ.
Hence we know that the expected number of iterations on input of size n is $1/\gamma(n)$, and each iteration clearly takes $T(n) + t(n)$.
Thus the expected time cost of the above procedure is $\frac{T(n)+t(n)}{\gamma(n)}$.

Problem 1.1

(a)

Let X_k denote the result of the kth coin toss with the biased coin and define the random variable

$$\tau = \min\{k \in \mathbb{N} \mid X_{2k-1} \neq X_{2k}\}$$

Now we define

$$Y = X_{2\tau-1}$$

We claim that the random variable Y represents a fair coin toss.

Indeed, by Baye's law and the independence of the biased tosses, we have for any $n \in \mathbb{N}$

$$\begin{aligned} P(X_{2n-1} = H \mid X_{2n-1} \neq X_{2n}) &= \frac{P(X_{2n-1} = H, X_{2n-1} \neq X_{2n})}{P(X_{2n-1} \neq X_{2n})} \\ &= \frac{P(X_{2n-1} = H, X_{2n} = T)}{P(X_{2n-1} \neq X_{2n})} \\ &= \frac{p(1-p)}{2p(1-p)} = 1/2 \end{aligned}$$

and likewise with tails.

To see that the expected number of tosses necessary to extract one fair toss with this procedure is $1/p(1-p)$ define

$$C_n = \begin{cases} 1, & \text{if } X_{2n-1} = X_n \\ 0, & \text{else} \end{cases}$$

Then $P(C_n = 1) = p^2 + (1-p)^2$ and $P(C_n = 0) = 2(p(1-p))$, so that if T denotes the number of tosses used we have for any $n \in \mathbb{N}$

$$P(T = 2n) = P(C_1 = 1, \cdots, C_{n-1} = 1, C_n = 0) = (p^2 + (1-p)^2)^{n-1} 2p(1-p)$$

So that the expected number of tosses is

$$\begin{aligned}
E(T) &= \sum_{n=1}^{\infty} 2n \left(p^2 + (1-p)^2\right)^{n-1} 2p(1-p) \\
&= 4p(1-p) \sum_{n=1}^{\infty} n \left(p^2 + (1-p)^2\right)^{n-1} \\
&= 4p(1-p) \frac{d}{dx} \left(\sum_{n=1}^{\infty} x^n \right) \Bigg|_{x=p^2+(1-p)^2} \\
&= 4p(1-p) \frac{1}{1-p^2-(1-p)^2} \\
&= 4p(1-p) \frac{1}{(2p(1-p))^2} \\
&= \frac{1}{p(1-p)}
\end{aligned}$$

as desired, where we have used that the power series converges for $|x| < 1$ and that $p^2 + (1-p)^2 < 1$ for $0 < p < 1$.

(b)

We first run the above described procedure on the list of biased flips until we reach the end of the list possibly disregarding the last flip if the number of flips is odd.
Every time we encounter a pair of different outcomes, we extract the unbiased coin flip as described and remove the pair from the list before we continue.
Now we are left with a list of pairs, where the outcomes of each pair are equal, and we create a new list by merging each pair into their common value.
Each entry in this new list is independent and distributed like the original flips, since the common value of a pair of tosses is determined by the value of the first. Hence we can run the procedure again on this new list, the length of which is at most half of that of the original list, extracting more unbiased coin tosses.
This procedure can be iterated until we are left with a list where all the values are equal, at which point we halt.

Problem 1.4

(b)

Consider two permutations $[i_1, \cdots, i_n]$ and $[j_1, \cdots, j_n]$. Then since the uniform variables are iid we have that the joint distribution of $(X_{i_1}, \cdots, X_{i_n})$ and $(E_{j_1}, \cdots, X_{j_n})$ are equal and hence

$$P\left(X_{i_1} < \cdots < X_{i_n}\right) = P\left(X_{j_1} < \cdots < X_{j_n}\right)$$

i.e., after sorting in ascending order, any two permutations of the variables are equally likely, hence they must all have probability $\frac{1}{n!}$ of occurring. So the indices of the sorted variables do indeed form a random permutation.

This permutation can be determined by sorting the random variables, e.g. by maintaining a list initialized to $[1, \cdots, n]$ and then performing the same operations on this list as on $[X_1, \cdots, X_n]$ when sorting it.

Treating the sampling of uniform variables as a black box, this scheme is as efficient as the sorting algorithm used to sort $[X_1, \cdots, X_n]$.

(c)