Cambridge Books Online

http://ebooks.cambridge.org/

Communication Complexity

Eyal Kushilevitz, Noam Nisan

Book DOI: http://dx.doi.org/10.1017/CBO9780511574948

Online ISBN: 9780511574948

Hardback ISBN: 9780521560672

Paperback ISBN: 9780521029834

Chapter

Answers to Selected Problems pp. 168-175

Chapter DOI: http://dx.doi.org/10.1017/CBO9780511574948.017

Cambridge University Press

Answers to Selected Problems

In this appendix we provide solutions for some of the exercises given throughout this book.

Solution for Exercise 1.7: We start by giving an $O(\log^2 n)$ bit solution for the median problem, MED, which is different than the one given in Example 1.6. Then we modify this solution to get the improved communication complexity. Assume, without loss of generality, that x and y have the same size and that this size is a power of 2 (they can reach this situation by exchanging the sizes of their inputs ($O(\log n)$ bits) and padding them with the appropriate numbers of the minimal element (1) and the maximal element (n)). The protocol works in stages. During the protocol Alice maintains a set x' consisting of all elements in her input set x that may still be the median (initially, $x' = x$) and Bob maintains a set $y' \subseteq y$ of all elements in y that may still be the median (initially, $y' = y$). At each stage, Alice sends Bob the value a, which is the median of x', and Bob sends the value b, which is the median of y'. If $a < b$, then Alice omits from x' the half smallest numbers and Bob omits from y' the half largest numbers. If $b < a$, then Alice omits from x' the half largest numbers and Bob omits from y' the half smallest numbers. Note that in both cases this omission maintains the median of $x' \cup y'$ and that the size of $x' \cup y'$ is reduced by a factor of 2. If $a = b$, then this value is obviously the median, and if $|x'| = |y'| = 1$, then the smaller number is the median. To conclude, the number of stages is $O(\log n)$ and in each of these stages the number of bits exchanged is $O(\log n)$ so the communication complexity is $O(\log^2 n)$.

To reduce the communication complexity to $O(\log n)$, we make two observations. First, in each stage the players only need to know which of a and b is the larger. For this, it is enough that they will exchange these numbers in a bit-by-bit manner (starting from the most significant bit) and stop immediately when they find a coordinate j in which $a_j \neq b_j$. Second, note that if a and b were the medians in a certain stage and, say, $a < b$, then in the next stage (and on) the remaining numbers in $x' \cup y'$ are in the range between a and b. Therefore, all the medians exchanged in the next stages must agree with the coordinates in which a and b agreed (that is, $1, \ldots, j-1$) and therefore the values in these coordinates need not be sent. This implies that each of the $\log n$ coordinates will not be exchanged more than once and the communication complexity of the modified protocol is therefore $O(\log n)$.

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

Solution for Exercise 1.8: For every graph G, we describe an appropriate protocol $\mathcal{P}_G$. The protocol works in stages, where the number of stages is at most $\log n$ and in each stage $O(\log n)$ bits are exchanged. All together this proves that $D(CIS_G) = O(\log^2 n)$. The players maintain a set of vertices $V' \subseteq V$ such that if C and I intersect, then the intersection is in V'. Initially $V' = V$. In each stage the players do the following:

- Alice looks for a vertex u in $C \cap V'$ such that the number of vertices in u's neighborhood (that is, u and the vertices adjacent to it according to G) that also belong to V' is at most $|V'|/2$. She sends the bit "0" if no such vertex exists; otherwise, she sends the bit "1" followed by the name of u ($\log n$ bits).
- If Alice sends a name of a vertex u, then Bob checks whether $u \in I$. If so, he outputs "1" (that is, there is an intersection). Otherwise, both players update V' (in their own memories) to be the intersection of V' with the neighborhood of u and they start a new stage. If Alice does not find a vertex u as needed, then Bob looks for a vertex w in $I \cap V'$ such that the number of vertices in w's neighborhood that also belong to V' is greater than $|V'|/2$. If there is no such vertex, Bob outputs "0" (that is, C and I are disjoint) and the protocol terminates. If there is such a vertex, w, then Bob sends its name to Alice.
- Alice checks whether $w \in C$. If so, she outputs "1" (that is, there is an intersection). Otherwise, both players update V' so that it contains only w and its nonneighbors that appeared in V' before the current stage.

For the correctness of the protocol, we claim that if C and I intersect in some vertex v then, by induction, $v \in V'$. This obviously holds when the protocol starts. There are two places in which V' is modified: (1) If Alice sends a name of a vertex u that is different than v. In this case, because C is a clique, v is a neighbor of u, and because it was in V' before the stage started (by the induction hypothesis) it also remains in V' when it is modified. (2) If Bob sends a name of a vertex w that is different than v. In this case, because I is an independent set, v is a nonneighbor of u, and because it was in V' before the stage started it also remains in V' when V' is modified. If the protocol outputs "1" then C and I clearly intersect. By the above claim, because the neighborhood of all vertices in $C \cap V'$ is of size at most $|V'|/2$ and the neighborhood of all vertices in $I \cap V'$ is of size greater than $|V'|/2$, then C and I are disjoint (otherwise, the intersection vertex v has neighborhood of size at most $|V'|/2$ and greater than $|V'|/2$ at the same time).

As for the complexity, it follows immediately from the protocol that the size of V' decreases by a factor of 2 in each stage and hence the number of stages is at most $\log_2 n$, and the communication complexity is $O(\log^2 n)$ as desired.

Solution for Exercise 2.18: The proof utilizes the *probabilistic method* (for an introduction to the probabilistic method the reader is referred to [Alon and Spencer 1992]). Choose at random a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. By this, we mean that for every input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ the value $f(x,y)$ is chosen to be 0 or 1 each with probability 1/2. In addition, the choices for different input pairs are independent. The expected number of 1s in f is $2^{2n}/2$. By Chernoff inequality, the probability that f has less than $2^{2n}/4$ 1s can be bounded by $\epsilon_1 = 2e^{-n/8}$. Now we show that the probability that $N^1(f) = \Omega(n)$ is large. For this it is enough to show that with high probability f has no 1-monochromatic rectangle of size $10 \cdot 2^n$ (obviously, this also implies that there are no larger monochromatic 1-rectangles). In such a case, to cover the 1s of f at least $\frac{2^{2n}}{4 \cdot 10 \cdot 2^n} = \frac{2^n}{40}$ 1-monochromatic rectangles are needed and therefore $N^1(f) = \Omega(n)$.

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

To prove this, note that the probability of a particular rectangle of size $10 \cdot 2^n$ to be 1-monochromatic is $2^{-10 \cdot 2^n}$ and that the number of rectangles is clearly bounded by $2^{2^n} \cdot 2^{2^n} = 2^{2 \cdot 2^n}$. Therefore, the probability that there exists a 1-monochromatic rectangle of size $10 \cdot 2^n$ is at most $\epsilon_2 = 2^{2 \cdot 2^n} \cdot 2^{-10 \cdot 2^n} = 2^{-8 \cdot 2^n}$.

Next, we prove that with high probability f has no fooling set of size $10n$ (therefore clearly, f cannot have a larger fooling set). The probability of a particular set of size $10n$ to be a fooling set is not larger than 2 (the number of choices for the value b that f gets on the fooling set pairs) times 2^{-10n} (the probability that for all pairs in the fooling sets b is chosen as the value of f) times $(3/4)^{\binom{10n}{2}}$ (the probability that, for each two distinct pairs (x_1, y_1) and (x_2, y_2) in the fooling set, for at least one of (x_1, y_2) and (x_2, y_1) the complement value $\bar{b}$ was chosen). This is bounded by 2^{-50n^2}. Now the number of possible fooling sets of size $10n$ is $\binom{2^{2n}}{10n} < 2^{20n^2}$. All together, the probability that a fooling set of size $10n$ exists for f is bounded by $\epsilon_3 = 2^{20n^2} \cdot 2^{-50n^2} = 2^{-30n^2}$.

To conclude, with very high probability (that is, at least $1 - \epsilon_1 - \epsilon_2 - \epsilon_3$) a randomly chosen function f satisfies $N^1(f) = \Omega(n)$ and it has no fooling set of size $10n$.

Solution for Exercise 3.18: Assume that $x \neq y$ (as shown in Example 3.13, this can be checked with $O(1)$ communication in the public coin model). The idea is that the two players will do a binary search for i_0, the most significant bit in which x and y differ. This bit i_0 clearly determines which of the numbers, x or y, is larger. We start by an $O(\log n \log\log n)$ solution and then modify it to get an $O(\log n)$ solution.

Suppose that Alice and Bob have a subprotocol $TEST(i,j)$ ($i \leq j$) that tests whether the two substrings $x_i, \ldots, x_j$ and $y_i, \ldots, y_j$ are equal. The players maintain two borders i and j ($i \leq j$) for the search (initially, $i = 1$ and $j = n$). In each stage, if $i = j$, then the players exchange their i-th bits to see which number is larger. If $i < j$, then the players define $m = (i+j)/2$ and run the subprotocol $TEST(i,m)$. If the result of this subprotocol is "not equal," they set $j = m$, whereas if the result is "equal," they set $i = m + 1$. In each case the difference $j - i$ is divided by 2 and hence the number of stages is $\log n$. The only question remaining is the implementation of $TEST(i,j)$. Example 3.13 shows that equality can be tested in the public coin model with $O(1)$ bits and error probability $1/4$. Thus if we implement $TEST$ by repeating this protocol $O(\log\log n)$ times we can reduce the error probability to, say, $1/(4\log n)$ while increasing the communication to $O(\log\log n)$. All together, because we use $TEST$ $\log n$ times, the communication complexity is only $O(\log n \log\log n)$ and with probability at most $\log n \cdot 1/(4 \log n) = 1/4$ any of them is wrong.

To improve the communication complexity, we use a more efficient $TEST$, with $O(1)$ bits (as in Example 3.13), but take into account that because $TEST$ fails with probability $1/4$ we may have to "fix" the search borders from time to time. However, because the protocol of Example 3.13 makes only one-sided error (that is, the only type of error is that it may say "equal" when actually the numbers are not equal), when j is updated then indeed $i_0 \leq j$, and only when i is updated an error can be made (that is, $i_0 < i$). The following protocol modifies the previous one by "fixing" the value of i from time to time.

1. Both players set $i = 1$ and $j = n$.
2. If $c \log n$ bits were already exchanged, then if $i < j$ stop with an arbitrary output (say, 0); otherwise, if $i = j$, Alice sends the bit x_i to Bob, who compares it with y_i and outputs 0 or 1 accordingly. If less than $c \log n$ bits were exchanged so far, then the players continue to the next step.

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

3. (FIXING) The players run $TEST(1, i-1)$. If the output is "equal" they proceed to the next step. If the output is "not equal" (this implies that $i_0 < i$) the players backtrack to the previous value of i and return to Step (2).

4. (SEARCH) If $i = j$, the players go to Step (2). Otherwise, if $i < j$, the players set $m = (i + j)/2$ and run $TEST(i,m)$. If the output is "equal" (this may be wrong), they store the current value of i and set $i = m + 1$ and if the output is "not equal," they set $j = m$. In both cases they return to Step (2).

The complexity is easy, since the protocol is forced to stop (Step (2)) if more than $c \log n$ bits are exchanged. For the correctness, we need to prove that within this many stages the players indeed find (with high probability) the bit i_0. Consider the sequence of pairs (i, j) the players have in Step (2) of the protocol (between any two executions of Step (2) the subprotocol *TEST* is used at most twice, hence each pair in the sequence corresponds to $O(1)$ bits of communication). A pair (i, j) in this sequence is *good* if i_0 is within the search borders (that is, $i \le i_0 \le j$) and, in addition, this is the first time that this pair appears in the sequence. As before, the difference $j - i$ is reduced by a factor of 2 each time that the borders are updated correctly. Hence, the number of good pairs is $\log n$. We will prove that the *expected* number of tests done between two consecutive *good* pairs is $O(1)$ and hence, by linearity of the expectation, an *expected* number of $O(\log n)$ tests is sufficient. This implies that stopping after $c \log n$ bits is also sufficient (with a constant increase in the error probability).

Consider a good pair (i, j). If i_0 is larger than $m = (i + j)/2$, then $TEST(i,m)$ always outputs "equal" and with a single test the players proceed to the next good pair. If however $i_0 \le m$, then with probability $3/4$ $TEST(i,m)$ outputs "not equal" and with a single test we are done, and with probability $1/4$ it outputs "equal" and we have a bad pair (i, j). In such a case, again, we have a probability $3/4$ of fixing i and probability $1/4$ of going deeper in the search (note that when we proceed with the wrong i any further updates of j still satisfy $i_0 \le j$ and any further updates of i still satisfy $i_0 < i$). In other words, at each step, with probability $3/4$ we take the right move and with probability $1/4$ we take the wrong move. We are interested in the expected number of steps until the number of right moves becomes larger than the number of wrong moves (at this point we will be in the next good pair). The probability that we will make exactly k steps is smaller than the probability that we will not finish within the first $k - 1$ steps. By Chernoff inequality, this probability is smaller than α^{k-1} for some constant $\alpha < 1$. Hence the expected number of steps needed is at most $\sum_{k=1}^{\infty} \alpha^{k-1} \cdot k$. This sum, by simple calculation, equals $(1/(1-\alpha))^2 = O(1)$.

Solution for Exercise 4.8 (Part (1)): A linear program is just an optimization problem that consists of a collection of linear inequalities that a solution should satisfy and an objective function to be maximized (or minimized). More precisely, let A be a $k \times \ell$ real matrix, let $\vec{b}$ be a length ℓ real vector, and let $\vec{c}$ be a length k real vector. Then a linear program has the form:

$$\max_{\vec{v}=v_1,\dots,v_\ell} \vec{b} \cdot \vec{v} \qquad \text{such that} \quad A\vec{v} \le \vec{c},\ \vec{v} \ge \vec{0}.$$

(In fact the definition can be made even more general but this form is enough for our purposes.) The duality theorem of linear programming states that the solution for this program (that is, the maximum value $\vec{b} \cdot \vec{v}$ as above) equals the solution of the following

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

minimization problem (sometimes referred to as the *dual* program):

$$\min_{\vec{u}=u_1,\ldots,v_k} \quad \vec{c}\cdot\vec{u} \qquad \text{such that} \quad A^T\vec{u} \geq \vec{b},\, \vec{u} \geq \vec{0}.$$

(For a background on linear programming see, for example, [Karloff 1991].)

To represent $B^1_*(f)$ by a linear program, let k be the number of 1-monochromatic rectangles with respect to f, and let ℓ be the number of input pairs (x, y) such that $f(x,y) = 1$. Let A be a $k \times \ell$ matrix such that each row corresponds to a 1-monochromatic rectangle R and each column corresponds to an input (x,y) such that $f(x,y) = 1$. The entry $(R,(x,y))$ of the matrix A equals 1 if (x,y) belongs to R and 0 otherwise. Consider the following linear program:

$$\max_{\vec{v}=v_1,\ldots,v_\ell} \quad \vec{1}\cdot\vec{v} \qquad \text{such that} \quad A\vec{v} \leq \vec{1},\, \vec{v} \geq \vec{0}.$$

We claim that α, the optimal solution for this program, equals $B^1_*(f)$ (it can be easily seen that $\alpha \geq 1$). On one hand, if $\vec{v}$ is a vector for which α is obtained, we can define a probability distribution μ on the 1s of f by $\mu(x,y) = \frac{v_{(x,y)}}{\alpha}$. Because $\vec{v}$ satisfies $\vec{v} \geq \vec{0}$ then, for every (x,y), $\mu(x,y) \geq 0$. Moreover, $\sum_{(x,y)} \mu(x,y) = \sum_{(x,y)} \frac{v_{(x,y)}}{\alpha} = \frac{\alpha}{\alpha} = 1$. Therefore μ is indeed a probability distribution. In addition, for every 1-monochromatic rectangle R, by the fact that $A\vec{v} \leq \vec{1}$ it follows that $\mu(R) \leq \frac{1}{\alpha}$. Thus, $B^1_*(f) \geq B_\mu(f) \geq \alpha$.

On the other hand, let μ be probability distribution on the 1s of f such that $B^1_*(f) = B_\mu(f)$. Define a vector $\vec{v}$ by $v_{(x,y)} = B_\mu(f)\cdot\mu(x,y)$. Because μ is a probability distribution, $\vec{v} \geq \vec{0}$. In addition, for every 1-monochromatic rectangle R, when multiplying the R row of A by $\vec{v}$ we get $B_\mu(f)\cdot\mu(R)$, which is at most $\frac{B_\mu(f)}{B_\mu(f)} = 1$, so $A\vec{v} \leq \vec{1}$. Therefore, $\alpha \geq \vec{1}\cdot\vec{v} = \sum_{(x,y)} B_\mu(f)\cdot\mu(x,y) = B_\mu(f) = B^1_*(f)$. All together, $\alpha = B^1_*(f)$.
Finally, we can write the corresponding dual program:

$$\min_{\vec{u}=u_1,\ldots,v_k} \quad \vec{1}\cdot\vec{u} \qquad \text{such that} \quad A^T\vec{u} \geq \vec{1},\, \vec{u} \geq \vec{0}.$$

Solution for Exercise 5.10: For the lower bound, we use Lemma 5.9 with X restricted to the set of all strings x such that $\#_1(x) = (n/2)+1$ (where, for a string w, $\#_1(w)$ denotes the number of 1s in w) and Y restricted to the set of all strings y such that $\#_1(y) = n/2$. For these sets X and Y, the set C satisfies $|C| = |Y|\cdot(n/2)$ (for every $y \in Y$ every 0-bit that is changed to 1 gives $x \in X$) and also $|C| = |X|\cdot((n/2)-1)$ (for every $x \in X$ every 1-bit that is changed to 0 gives $y \in Y$). Hence

$$C^D(R_{\text{MAJ}}) \geq \frac{|C|^2}{|X||Y|} = \frac{|Y|\cdot(n/2)\cdot|X|\cdot((n/2)-1)}{|X||Y|} = n^2/4 - n/2.$$

That is, $D(R_{\text{MAJ}}) \geq 2\log n - O(1)$.

For the upper bound, we start with an $O(\log^2 n)$ protocol. At stage $i+1$ Alice views her current string $x(i)$ as consisting of two (equal-sized) halves x_L and x_R (that is, $x(i) = x_L \circ x_R$). She sends Bob $\#_1(x_L)$ and $\#_1(x_R)$ ($O(\log n)$ bits). Bob compares these numbers with $\#_1(y_L)$ and $\#_1(y_R)$. He tells Alice (using a single bit) in which half the numbers of 1s differ (if this holds in both halves he chooses one of them arbitrarily). Alice and Bob set $x(i+1)$ and $y(i+1)$ (respectively) to this half of their strings. By induction, there is always a half with different number of 1s (because we start with $\#_1(y) \leq n/2 < \#_1(x)$). After $\log n$ stages, the players have substrings of length 1 in which x and y differ. The index of this bit is the desired output. The total communication is $O(\log^2 n)$.

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

To get an $O(\log n)$ protocol, we modify the above protocol as follows. In the above protocol, the only aim was to make sure that $\#_1(x(i)) \neq \#_1(y(i))$. Here, Alice and Bob at the end of the i-th stage will also have a number d_i such that $|\#_1(x(i)) - \#_1(y(i))| \geq 2^{d_i}$ $(0 \leq d_i \leq \log n)$. The intuition is that if d_i is large the players probably sent many bits so far but will be able to save bits in the next stages. The $i+1$ stage of the protocol goes as follows. We know that in (at least) one of the halves, say the left, the difference in the number of 1s is at least 2^{d_i-1} (the players do not know which of the halves is this, so they will do the following in each of the two halves in parallel). Let $a_{\log n} \cdots a_2 a_1$ be the binary representation of $\#_1(x_L)$ and $b_{\log n} \cdots b_2 b_1$ the binary representation of $\#_1(y_L)$. The players exchange these numbers in a bit-by-bit manner, going from the least significant bits toward the most significant bits. However, they start from position $d_i - 1$, ignoring the $d_i - 2$ least significant bits in positions $1, \ldots, d_i - 2$ (when d_i is very small we may refer to positions that are smaller than 1; think of this positions as containing 0s). If $\#_1(x_L)$ and $\#_1(y_L)$ differ by at least 2^{d_i}, this implies that the truncated numbers (i.e, $a' = a_{\log n} \ldots a_{d_i-1}$ and $b' = b_{\log n} \ldots b_{d_i-1}$) differ by at least 2. The players stop exchanging their bits in a position ℓ according to these rules:

1. If $a_{d_i-1} = b_{d_i-1}$, then ℓ is the first position in which the bits differ (there must be such a position because $a' \neq b'$). That is, $a_\ell \neq b_\ell$ and for all $d_i - 1 \leq \ell' < \ell$, $a_{\ell'} = b_{\ell'}$.
2. If $a_{d_i-1} \neq b_{d_i-1}$ but $a_{d_i} = b_{d_i}$, then ℓ is the next position in which the bits differ (there must be such a position because a' and b' differ by at least 2).
3. If both $a_{d_i-1} \neq b_{d_i-1}$ and $a_{d_i} \neq b_{d_i}$, then if in one of the numbers both bits are 1s and in the other number both are 0s, stop in position $\ell = d_i$. Otherwise, the bits in one of the numbers are 10 and in the other 01. Again, because a' and b' differ by at least 2 (and $10 - 01$ is 1), we let ℓ be the next position in which the bits differ.

Let $k_{i+1} = \ell - d_i + 1$ and $d_{i+1} = d_i + k_{i+1} - 3$. The number of bits transmitted in stage $i+1$ is $4k_{i+1}$ (Alice and Bob exchange the positions $d_i - 1, \ldots, \ell$ in the 4 numbers $\#_1(x_L)$, $\#_1(x_R)$, $\#_1(y_L)$, and $\#_1(y_R)$). We now prove that the difference at the end of stage $i+1$ is indeed at least $2^{d_{i+1}}$. Because the proof for all stopping rules is similar we prove it only for case (1), leaving the details of the other cases to the reader. Assume, without loss of generality, that $a_\ell = 1$ and $b_\ell = 0$. There are two subcases depending on the bits in positions $\ell + 1, \ldots, \log n$. Let β_a be the number represented by $a_{\log n} \ldots a_{\ell+1}$ and β_b represented by $b_{\log n} \ldots b_{\ell+1}$. If $\beta_a \geq \beta_b$, then $a - b \geq 2^{\ell-1} - (2^{d_i-1} - 1)$, where the term $2^{\ell-1}$ is contributed by the ℓ-th position (and the more significant bits are either equal or larger in a which makes the difference only bigger) and we subtract $2^{d_i-1} - 1$ because we do not know anything about the content of positions $1, \ldots, d_i - 1$. Because $\ell \geq d_i + 1$ we get $a - b > 2^{\ell-2} = 2^{d_i+k_{i+1}-3}$. The second subcase is where $\beta_a < \beta_b$. In this case, using similar arguments,

$$b - a \geq (\beta_b - \beta_a) \cdot 2^\ell - 2^{\ell-1} - (2^{d_i-1} - 1) \geq 2^{\ell-1} - (2^{d_i-1} - 1) \geq 2^{d_i+k_{i+1}-3}.$$

We showed that $4 \cdot k_{i+1}$ bits are sent in the $i+1$ stage and $d_{i+1} = d_i + k_{i+1} - 3$. This implies that the total communication is

$$4 \cdot \sum_{i=0}^{\log n-1} k_{i+1} = 4 \cdot \sum_{i=0}^{\log n-1} ((d_{i+1} - d_i) + 3) = 4 \cdot (3 \log n + d_{\log n} - d_0) = O(\log n).$$

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

Solution for Exercise 7.11 (Part (2)): The idea is to use the technique of Example 7.9 (the "shifted equality" function, SEQ) by embedding the "shifted inner product" function into the function PROD as follows. Let $k = \log n$, $t = n/k$, and let $0 \le s,r < k$ be two parameters (to be chosen). Given $x,y \in \{0,1\}^t$, we define $a,b \in \{0,1\}^n$ by assigning $a_{ki+s} = x_i$ and $b_{kj+r} = y_j$, for all $0 \le i,j < t$, and fix all other entries of a and b to zero (that is, most of the entries of a and b contain 0s; and the 1s are separated by at least $k = \log n$ 0s). It can be verified that for $0 \le \ell < 2t$,

$$\text{PROD}(a,b,k\ell + s + r) = \sum_{i=0}^{\ell} x_i y_{\ell-i} \bmod 2 = \langle x_0 \dots x_\ell, y_\ell \dots y_0 \rangle$$

(where for $i \ge t$, x_i and y_i are defined to be zero). This uses the fact that due to the sparseness of 1s in a and b there is no carry for more than k steps.

There are two minor technical problems to solve before we can emulate the proof of Example 7.9 using the lower bound for IP (instead of the lower bound for EQ). First, the bits of x and y are only a fraction of $1/\log n$ of the bits of a and b. Hence, in principle, it is possible that all of them are given to one player. Second, because the length of the inner product induced by a certain value of ℓ is $\ell + 1$, there is no complete symmetry for all shifts (values of ℓ). We overcome these difficulties as follows: as in the case of SEQ we may assume, without loss of generality, that Alice holds at least $n/2 - O(\log n)$ bits of a and Bob holds at least $n/2 - O(\log n)$ bits of b. Now we use the freedom to choose s and r. We look at Alice's bits and fix s to be such that Alice holds the largest number of bits of the form a_{ki+s}. Denote by A the set of $0 \le i < t$ such that Alice holds the bit a_{ki+s}, and notice that $|A| \ge \frac{1}{k}(n/2 - O(\log n)) = \Omega(t)$. Similarly, fix r to be such that Bob holds the largest number of bits of the form b_{kj+r}. Denote by B the set of $0 \le j < t$ such that Bob holds the bit b_{kj+r} (again, $|B| = \Omega(t)$). Fix all bits outside of A and B to zero.

Let $B_\ell = \{i \mid \ell - i \in B\}$. We now claim that there is a value for ℓ that satisfies $|A \cap B_\ell| = \Omega(t) = \Omega(m/\log m)$. This will conclude the proof due to the lower bound on the deterministic communication complexity of IP (Example 1.29). This again can be proven as in Example 7.9. We use here a somewhat different argument (which is essentially the same proof with a different terminology): consider the expected value for $|A \cap B_\ell|$, where ℓ is chosen at random in $0,\dots,2t-1$. For an arbitrary $i \in A$, the probability that $0 \le \ell - i < t$ is at least 1/2. Conditioned on this happening, ℓ is uniformly distributed in $0,\dots,t-1$, and thus $\Pr[i \in B_\ell] \ge \alpha$, for some constant α (recall that $|A|$ and $|B|$ are $\Omega(t)$). It follows that the expected value for $|A \cap B_\ell|$ is $\Omega(t)$ and thus some value of ℓ achieves at least this.

Solution for Exercise 7.15: Consider the graph G_A consists of the n edges that are input to Alice and the graph G_B consists of the n edges that are input to Bob. Let $d = n/4\ell$. Let S_A be a set of d edges $(u_1,v_1),\dots,(u_d,v_d)$ in G_A such that $v_1,\dots,v_d$ are all distinct and they are different from $u_1,\dots,u_d$ ($u_1,\dots,u_d$ may not be distinct). Such a set of edges must exist because either there is a vertex u in G_A of degree at least d (in which case S_A is a set of d edges connected to u), or else, if all vertices are of degree at most $d-1$, the graph G_A contains a matching of size at least d (in which case S_A is a matching of size d). The reason is that we can construct such a matching in a greedy way: at each step we pick an edge $e = (u,v)$, add it to the matching, and eliminate the edge e together with all other edges connected to u and v (at most $d-2$ additional edges for each of the two vertices). All together, at most $2d-3$ edges. Because $d(2d-3) \le n$ this can be repeated at least d times. Now consider the graph G_B. Remove from it all edges

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015

that have a joint vertex with any of the d edges in S_A. Because there are at most $2d$ vertices in S_A, the number of edges eliminated from G_B is at most $2d \cdot \ell \leq n/2$. In other words, in the modified graph G'_B, there are at least $n/2$ edges. Therefore, in a similar way, we can find in G'_B a set S_B of d edges $(u'_1,v'_1),\ldots,(u'_d,v'_d)$ such that $v'_1,\ldots,v'_d$ are all distinct and they are different from $u'_1,\ldots,u'_d$. (By the definition of G'_B, the vertices $u'_1,\ldots,u'_d,v'_1,\ldots,v'_d$ are all different from $u_1,\ldots,u_d,v_1,\ldots,v_d$.)

We now fix all input bits (that is, edges) not in S_A or S_B. We will show that the disjointness function DISJ on $\{0,1\}^d \times \{0,1\}^d$ can be reduced to the communication problem defined by S_A and S_B. This implies, by Example 1.23, that at least $d = \Omega(n/\ell)$ bits are required to solve USTCON, as needed. Connect every two distinct vertices u_i and u_j with an edge (that is, fix the input bit corresponding to the edge (u_i,u_j) to 1). Similarly, connect every two distinct vertices u'_i and u'_j with an edge. Also, connect vertex s of the graph to $u_1,\ldots,u_d$ and vertex t of the graph to $u'_1,\ldots,u'_d$ and put an edge connecting v_i to v'_i, for all $1 \leq i \leq d$. All other edges out of S_A and S_B do not exist (that is, fix the corresponding input bits to 0). The reader may now verify that vertices s and t of the graph are connected if and only if for some j the edge (u_j,v_j) (given to Alice) and the edge (u'_j,v'_j) (given to Bob) are both set to 1. Hence the problem that Alice and Bob need to solve in this case is the disjointness function.

Downloaded from Cambridge Books Online by IP 59.78.20.61 on Sun Sep 20 03:29:55 BST 2015.
http://dx.doi.org/10.1017/CBO9780511574948.017
Cambridge Books Online © Cambridge University Press, 2015