Final Solutions

(1) The ϵ transition allows us to go from the start state to B. Thus, 01 is accepted (going from A to B to C to B), but 00 is not, as the only way to process the initial 0 is to go from B to C and then we are stuck.

(2) We assume that the invariant is true before the loop starts. If x is even, then the product $x \cdot y$ does not change and neither does total, so the invariant still holds. If x is odd, then we can write it as $x = 2 \cdot z + 1$. Before the loop, we have $x \cdot y +$total $= a \cdot b$. When we are done with the loop, we have $x \leftarrow z,\ y \leftarrow 2 \cdot y,$ total $=$ total$+y$. In terms of the new values, we need to look at $x \cdot y+$total, which in terms of the old values is $z \cdot 2 \cdot y+$total$+y$ which simplifies to $(z \cdot 2 \cdot y + y)+$total $= (2z+1) \cdot y+$total $= x \cdot y+$total and is therefore the same value. The algorithm stops if $x == 0$, in which case total $= a \cdot b$.

(3) The new algorithm has recurrence $T(n) = 23T(n/5) + c$ with asymptotic run-time $\Theta(n^{\log_5(23)})$. Because $\log_5(23) > \log_2(3)$, there is no gain over the original.

(4) Write $n = 3m + 2$. For the m groups of threes, we use three comparisons each for a total of $3m$ and for the small group we use one comparison in order to obtain the minima and maxima of all groups. So far, we have used $3m + 1$ comparisons. To obtain the maxima among the $m + 1$ group maxima, we use m comparisons and the same for the minima. This step gives an additional $2m$ comparisons. All together, we have $5m + 1$ comparisons. For the naïve algorithm, we have $2(n-1) = 2(3m+2-1) = 6m+3$ comparisons, which is more.

(5) $50 = 32 + 18 = 2^5 + 18$. Therefore, the split pointer is 18 and the level is 5.

(6) We number the elements in the set as $x_1, x_2, \ldots, x_n$. Let $f(r,t)$ be the closest we can get to t using the first r elements, i.e. the elements in $x_1, x_2, \ldots, x_r$. Then $f(r+1,t) = \min(f(r,t), f(r, t - x_{r+1})$.

(7) We number the elements in the sequence as $x_1, x_2, \ldots, x_n$. We again consider the sequence of sequences up to the r-th element. For each r, we maintain **two** sequences: The longest strictly increasing sequence in $x_1, x_2, \ldots, x_r$ and the longest strictly increasing sequence that ends in x_r. When we increment r, we update the two sequences depending on whether x_r is larger than x_{r-1} or not.

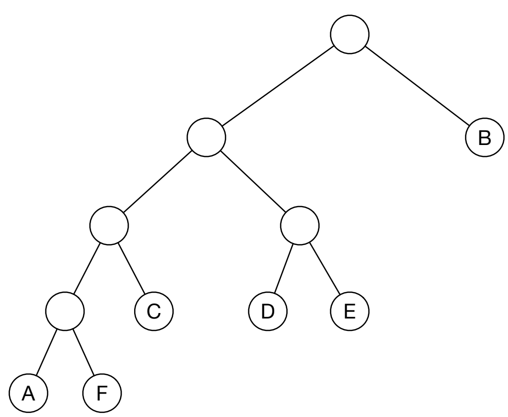
(8) The waypoint matrix tells us that we need to go via Vertex 2. This breaks the problem into getting to 2 and then getting from 2 to 4. For the first one, we get waypoint 1, for the second, we get waypoint 4, meaning going directly from 2 to 4. In total, we get 0 -> 1 -> 2 -> 4.

(9) We copy the table and annotate it:

		A	B	C	D	E	F	G	H	I	J
0 :	0	**0**	**0**	**0**	0	0	0	0	0	0	0
1 :	0	0	0	0	0	0	0	0	0	0	0
2 :	0	4	4	4	4	4	4	4	4	4	4
3 :	0	4	5	6	6	6	6	6	6	6	6
4 :	0	4	5	6	**9**	9	9	9	9	9	9
5 :	0	4	9	10	10	13	13	13	13	13	13
6 :	0	4	9	11	13	13	16	16	16	16	16
7 :	0	4	9	11	15	17	17	19	19	19	19
8 :	0	4	9	15	15	19	20	20	20	21	21
9 :	0	4	9	15	19	**22**	**22**	23	23	23	25
10 :	0	4	9	15	20	23	25	25	25	25	25
11 :	0	4	9	15	20	26	29	29	29	29	29
12 :	0	4	9	15	24	28	29	32	32	32	32
13 :	0	4	9	15	24	28	33	35	35	35	35
14 :	0	4	9	15	24	32	35	36	36	37	38
15 :	0	4	9	15	24	33	38	39	39	40	41
16 :	0	4	9	15	24	33	39	**41**	**41**	**41**	44
17 :	0	4	9	15	24	37	42	44	44	44	46
18 :	0	4	9	15	24	37	44	48	48	48	48
19 :	0	4	9	15	24	37	44	48	49	50	50
20 :	0	4	9	15	24	37	48	52	52	53	54
21 :	0	4	9	15	24	37	49	54	55	56	57
22 :	0	4	9	15	24	37	49	57	57	57	60
23 :	0	4	9	15	24	37	53	58	59	60	62
24 :	0	4	9	15	24	37	53	61	61	62	65
25 :	0	4	9	15	24	37	53	63	64	65	**66**

This means we include J, G, E, and D.

10. We first join A and F to have a super-symbol AF with frequency 13%. Then we join AF with C to get (AF)C with frequency 25%. Then we join DE for a combined frequency of 35%. We then join (AF)C and DE and afterwards with B.



A possible encoding is A — 0000, B —1, C — 001, D — 010, E — 011, F — 0001.

11. $\delta(s, v) \leq \delta(s, u) + w$.
12. In this case, v is an ancestor of u so that there is a path from v to u. The edge from u to v finishes a cycle.
13. Four nodes have degree 3 and therefore the graph is not Eulerian.
14. We have had to start out in A, and then discover B. We just finished B and the next step is to select the edge with smallest provisional distance from A, namely C.
15. Since no algorithm exists, *a fortiori* no poly-time algorithm exists for the halting problem.
16. Problem A is in NP. Assume any other problem in NP and assume that I can solve Problem A in poly-time. Then by assumption, I can solve Circuit-Satisfiability in poly-time. If I can solve Circuit-Satisfiability, then I can that other problem in poly-time. This means, Problem A is NP-complete.
17. There are c possibilities to color one edge. As there are e edges, there are c^e possible coloring. Any algorithm that does a proportion of this complete enumeration (as we have to assume back-tracking will do for most graphs) is not polynomial time. The problem is however in $\mathcal{NP}$ because checking a coloring can be done by looking at all the edges in all adjacency lists. We will look at v vertices and at each edge twice (because each edge appears in two adjacency lists) for a total of $\Theta(v) + \Theta(e)$ runtime.
18. A Hamiltonian path combines all v vertices, and therefore has to encompass $v - 1$ edges. If the number of edges is less than that, then no Hamiltonian path can exist.
19. See table below. On average, we shift by 2.25 characters if all characters in the text are equally likely to occur at any given position.

bad character	Shift
"T"	matches, continue
"A"	2
"C"	1
"G"	6