Given an algorithm, a configuration is an assignment of values to each variable. (In this discussion, the input values such as the weights of the edges in Dijkstra’s algorithm that remain constant throughout the algorithm are not variables.) A configuration is feasible if it can actually occur during the execution of the algorithm. Let $C$ denote the set of all configurations, whether feasible or not. We refer to $C$ as the configuration space.

A predicate over a set $A$ is a Boolean function $f : A \rightarrow \{0, 1\}$ (1: “true,” 0: “false”). A transformation of a set $A$ is a function $g : A \rightarrow A$.

Let $a \in A$. The following statements are synonymous:

- the predicate $f$ is true for $a$;
- the predicate $f$ holds for $a$;
- the element $a$ satisfies the predicate $f$;
- $f(a) = 1$.

Let $P$ and $R$ be predicates over the configuration space $C$ and let $T$ be a set of instructions, viewed as a transformation of $C$. Consider the loop

\[
\text{while } P \text{ do } T
\]

We say that $R$ is a loop invariant for this loop if for all configurations $X \in C$ the following inference is correct:

\[
P(X) \& R(X) \Rightarrow R(T(X)).
\]

In other words, if the configuration $X$ satisfies the predicate $R$ then the updated configuration $T(X)$, obtained by legally executing the loop instructions, also satisfies $R$. An execution of the loop instructions on a configuration $X$ is legal if $X$ satisfies the loop condition $P$.

Note that the highlighted statement (2) has to hold even for infeasible configurations. This is analogous to chess puzzles: when showing that a certain configuration leads to checkmate in two moves, you do not investigate whether or not the given configuration could arise in an actual game.

We shall see (esp. in Ex. 1 below) that a loop invariant $R$ acts as the inductive hypothesis in an inductive proof of the correctness of the algorithm and proving that $R$ is a loop invariant is the induction step of the proof.
Let $T^k(X)$ denote the configuration obtained from $X \in \mathcal{C}$ after executing the instruction set $T$ $k$ times; let $T^0(X) = X$.

**Exercise 1.** Prove: if $R$ is loop invariant for the loop (1) and $P(T^i(X))$ holds for all $i \leq k - 1$ and $R(X)$ holds then $R(T^k(X))$ holds. In other words, if the loop invariant is true for a configuration then it remains true after any number of legal executions of the loop instructions.

Now we consider how to build complex loop invariants out of simpler predicates.

**Exercise 2.** Prove: if $R_1$ and $R_2$ are loop invariants for the loop (1) then $R_1 \& R_2$ is also a loop invariant for (1).

**Relative loop invariants.** Let $R_1$ and $R_2$ be predicates over the configuration space. We say that $R_2$ is a loop invariant modulo $R_1$ if for all configurations $X \in \mathcal{C}$ the following inference is correct:

$$P(X) \& R_1(X) \& R_2(X) \Rightarrow R_2(T(X)).$$

(3)

In other words, we only require $R_2$ to satisfy Eq. (2) for those configurations $X$ that satisfy $R_1$. Other terms used to describe this circumstance: $R_2$ is a loop invariant assuming $R_1$ or relative to $R_1$.

**Exercise 3.** Prove: if $R_1$ is a loop invariant for the loop (1) and $R_2$ is a loop invariant modulo $R_1$ for (1) then $R_1 \& R_2$ is a loop invariant for (1).

**Analysis of Dijkstra’s algorithm.** Please consult the handout on the particular version of Dijkstra’s algorithm we analyze here.

A configuration for Dijkstra’s algorithm consists of a status value (white, grey, black), a cost value (a real number or $\infty$), and a parent link (possibly NIL) for each vertex, and a set $Q$ (the priority queue; here we treat it as a set of nodes; the key on which priority is based is the cost).

The body of Dijkstra’s algorithm (i.e., the part of the algorithm after the initialization of the variables) consists of iterations of a single “while” loop. The loop condition is “$Q \neq \emptyset$.” Below, loop invariance refers to this while loop.

Consider the following statements:

$R_1: \quad (\forall u \in V)(u \in Q \text{ if and only if } u \text{ is grey})$

$R_2: \quad (\forall u \in V)( \text{ if } u \text{ is white then } c(u) = \infty).$

$R_3: \quad (\forall u, v \in V)( \text{ if } u \text{ is black and } v \text{ is not black then } c(u) \leq c(v)).$

$R_4: \quad (\forall v \in V)(c(v) \text{ is the minimum weight among all } s \rightarrow \cdots \rightarrow v \text{ paths that pass through black vertices only}).$
(We say that the path \( s = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k = v \) passes through black vertices only if for \( 0 \leq i \leq k - 1 \), the vertex \( v_i \) is black. The vertex \( v \) may have any status. The weight of the path is the sum \( \sum_{i=1}^{k} w(v_{i-1}, v_i) \). If \( \mathcal{P} \) denotes this path then we write \( w(\mathcal{P}) \) to denote its weight.)

**Exercise 4.** Prove that \( R_1 \) and \( R_2 \) are loop invariants.

**Exercise 5.** Consider the following statement:

\[ R_2^* : \ (\forall u \in V)(u \text{ is white if and only if } c(u) = \infty). \]

Prove that (a) \( R_2^* \) is not a loop invariant; but (b) \( R_2^* \) is a loop invariant modulo \( R_1 \).

**Exercise 6.** Prove that \( R_3 \) is a loop invariant modulo \( R_1 \ & R_2 \).

Warning: As in class, this is a non-trivial proof; several cases need to be handled separately. The solution to this exercise has been added at the end of this document.

**Exercise 7.** Prove that \( R_4 \) is not a loop invariant modulo \( R_1 \ & R_2 \). (We need \( R_3 \).)

*Explanation.* You need to construct a weighted directed graph with nonnegative weights, a source, and an assignment of values to each variable (parent links, status colors, current cost values, priority queue) such that \( R_1 \ & R_2 \ & R_4 \) holds for your configuration and your configuration satisfies the loop condition \( Q \neq \emptyset \), but \( R_4 \) will no longer hold after executing Dijkstra’s **while** loop. Your graph should have very few vertices.

**Exercise 8.** Prove that \( R_4 \) is a loop invariant modulo \( R_1 \ & R_2 \ & R_3 \).

Warning: Again, as in class, this is a non-trivial proof; several cases need to be handled separately. The solution to this exercise has been added at the end of this document.

**Exercise 9.** Infer from the exercises above that Dijkstra’s algorithm is correct. (Of course Exercises 5 and 7 are not needed.)

This is a very simple exercise. The essence of the proof of Dijkstra’s correctness is in Exercises 6 and 8.
Solution to Exercise 6.

We need to analyze what happens during an execution of the while loop.

Every variable has an “in” and an “out” value; its value before the beginning of the current execution of the loop and its value after the end of the current execution of the loop. We shall omit the phrase “before the beginning/after the end of the current execution of the loop” and simply say that a statement is true “before” and/or “after.” We say something happens “currently” if it happens during the current execution of the loop.

Let $B_{\text{in}}$ denote the set of black vertices “before” and $B_{\text{out}}$ the set of black vertices “after.” Let $u$ denote currently explored vertex, i.e., the vertex extracted from $Q$ at the beginning of the current execution of the loop. So $\text{status}_{\text{in}}(u) = \text{grey}$ and $\text{status}_{\text{out}}(u) = \text{black}$, and $B_{\text{out}} = B_{\text{in}} \cup \{u\}$.

Let $w \in B_{\text{out}}$ and $v \in V \setminus B_{\text{out}}$. We need to show that $c_{\text{out}}(w) \leq c_{\text{out}}(v)$.

Observation A. $c_{\text{out}}(w) = c_{\text{in}}(w)$ and $c_{\text{out}}(u) = c_{\text{in}}(u)$.

Proof: The loop does not change the cost of any initially black vertex, nor does it change the cost of $u$. $\square$ (Obs. A)

Observation B. $c_{\text{in}}(w) \leq c_{\text{in}}(u)$.

Proof: Case B1. $w = u$. In this case $c_{\text{in}}(w) = c_{\text{in}}(u)$.

Case B2. $w \neq u$. In this case $w \in B_{\text{in}}$. On the other hand $u \notin B_{\text{in}}$. Therefore $c_{\text{in}}(w) \leq c_{\text{in}}(u)$ because $R_3$ was true “before.” $\square$ (Obs. B)

Observation C. $c_{\text{in}}(u) \leq c_{\text{in}}(v)$.

Proof: We know that $v \notin B_{\text{out}}$. It follows that $v \notin B_{\text{in}}$, i.e., $\text{status}_{\text{in}}(v) \neq \text{black}$. So we have two cases to consider.

Case C1: $\text{status}_{\text{in}}(v) = \text{white}$. Then $c_{\text{in}}(v) = \infty$ by $R_2$; therefore $c_{\text{in}}(u) \leq c_{\text{in}}(v)$.

Case C2: $\text{status}_{\text{in}}(v) = \text{grey}$. Then $v \in Q_{\text{in}}$ by $R_1$ and therefore $c_{\text{in}}(v)$ was one of the keys with which $c_{\text{in}}(u)$ was compared when $u$ was extracted, so again $c_{\text{in}}(u) \leq c_{\text{in}}(v)$. $\square$ (Obs. C)

Claim D. $c_{\text{in}}(u) \leq c_{\text{out}}(v)$.

Proof.

Case D1. $c_{\text{out}}(v) = c_{\text{in}}(v)$ (i.e., the cost of $v$ is not reduced during the current execution of the loop). In this case we are done by Observation C.

Case D2. $c_{\text{out}}(v) \neq c_{\text{in}}(v)$, i.e., the cost of $v$ is reduced during the execution of the loop. This means $(u, v) \in E$ and $c_{\text{out}}(v) = c_{\text{in}}(u) + w(u, v) \geq c_{\text{in}}(u)$, as desired. Note that the last inequality depended on the assumption that the weights are non-negative. $\square$ (Claim D)

Finally, putting all this together, we see that...
\( c_{\text{out}}(w) \overset{(A)}{=} c_{\text{in}}(w) \overset{(B)}{\leq} c_{\text{in}}(u) \overset{(D)}{\leq} c_{\text{out}}(v) \). This completes the proof of the desired inequality \( c_{\text{out}}(w) \leq c_{\text{out}}(v) \). \( \square \)  (Ex. 6)

**Exercise 10.** Where did this proof use

- \( R_1 \)
- \( R_2 \)
- the assumption that \( R_3 \) was true “before”
- that the edges have non-negative weights ?

Did the proof use Observation C ?

* * *

Solution to Exercise 8.

\( R_4 \) asserts two things. First, that

\( R_{4a} : \ (\forall v \in V)( \text{if } v \text{ is not white then there exists an } s \rightarrow \cdots \rightarrow v \text{ path } \mathcal{P} \text{ that passes through black vertices only, such that } c(v) = w(\mathcal{P}) ). \)

Second, it says

\( R_{4b} : \ (\forall v \in V)( \text{if } \mathcal{P} \text{ is an } s \rightarrow \cdots \rightarrow v \text{ path that passes through black vertices only then } c(v) \leq w(\mathcal{P}) ) \).

**Exercise 11.** Prove that \( R_{4a} \) is a loop invariant modulo \( R_1 \& R_2 \).

Hint: Prove that the following is a loop invariant modulo \( R_1 \& R_2 \).

\( R_{4c} : \ (\forall v \in V)( \text{if } v \neq s \text{ and } v \text{ is not white then } p(v) \text{ is black}) \).

So in order to solve Exercise 8, we only need to prove that \( R_{4b} \) is a loop invariant modulo \( R_1 \& R_2 \& R_3 \& R_{4a} \).

Consider a path \( s = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k = v \) that passes through black vertices only in the configuration “after.” The stipulation that this happens “after” allows \( \mathcal{P} \) to pass through \( u \), the vertex currently being explored.

**We need to show that** \( c_{\text{out}} \leq w(\mathcal{P}) \).

Proof.
Case 1: $u \notin \mathcal{P}$. In this case $c_{\text{in}}(v) \leq w(\mathcal{P})$ because $R_4$ is assumed to hold “before.” But $c_{\text{out}}(v) \leq c_{\text{in}}(v)$ (costs can only go down) so the desired inequality, $c_{\text{out}} \leq w(\mathcal{P})$, follows.

Case 2: $u \in \mathcal{P}$. For $i \leq j$, let $\mathcal{P}[v_i, v_j]$ denote the segment of $\mathcal{P}$ from $v_i$ to $v_j$.

Observation E. If $i \leq j \leq \ell$ then $w(\mathcal{P}[v_i, v_j]) = w(\mathcal{P}[v_i, v_j]) + w(\mathcal{P}[v_j, v_{\ell}])$.

Clear. No variables are involved in this statement. \text{□ (Obs. E)}

Observation F. $c_{\text{in}}(u) \leq w(\mathcal{P}[s, u])$.

Proof: Follows from the assumption of $R_4$ “before.” \text{□ (Obs. F)}

Let now $z$ denote the last vertex of $\mathcal{P}$ before reaching $v$.

Case 2a: $z = u$. In this case we have $(u, v) \in E$ (the last step of $\mathcal{P}$). Therefore $c_{\text{out}}(v) \leq c_{\text{in}}(u) + w(u, v) \leq w(\mathcal{P}[s, u]) + w(u, v)$ (E) $\leq w(\mathcal{P})$. \text{□ (2a)}

Case 2b: $z \neq u$. In this case $z \in B_{\text{in}}$ while $u \notin B_{\text{in}}$; therefore

$$c_{\text{in}}(z) \leq c_{\text{in}}(u) \quad (4)$$

by $R_3$. By $R_{4a}$, there is a path $T$ through $B_{\text{in}}$ such that $c_{\text{in}}(z) = w(T)$. Let $T^*$ be the $s \rightarrow \cdots \rightarrow z \rightarrow v$ path obtained by adding the $(z, v)$ edge to $T$.

Now

$$c_{\text{out}}(v) \leq w(T^*) \quad (5)$$

because $u \notin T^*$ so this case falls under Case 1.

On the other hand,

$$w(T) = c_{\text{in}}(z) \leq c_{\text{in}}(u) \quad (4) \quad R_{4b} \quad \text{before} \quad \leq w(\mathcal{P}[s, u]) \leq w(\mathcal{P}[s, z]), \quad (6)$$

where the last inequality follows from Obs. E and the non-negativity of the edge weights. Finally,

$$c_{\text{out}}(v) \leq w(T^*) = w(T) + w(z, v) \leq w(\mathcal{P}[s, z]) + w(z, v) = w(\mathcal{P}),$$

justifying the $c_{\text{out}}(v) \leq w(\mathcal{P})$ claim. \text{□ (Case 2b)}

This completes the proof of the loop invariant $R_{4b}$ and thereby $R_4$ (modulo $R_1 \& R_2 \& R_3 \& R_{4a}$). \text{□ (Ex. 8)}

**Exercise 12.** Ask and answer the questions analogous to Exercise 10 regarding this proof.