Flooding

Flooding is the process of raising the terrain by uniformly pouring water onto it until all sinks are filled and a steady-state (where there is a non-increasing flow path from each cell to the edge of the terrain) is reached. After flooding, we say that a watershed u has been *raised* to height h if every cell in u lower than h is raised to height h. Let G_T be the watershed graph of a terrain T and let the *height of a path p* in G_T be the maximum weight of the edges along p. We can formally define flooding as follows.

Definition 1 ((Flooding)) *Flooding of a terrain T is the process of raising each watershed $u \neq \zeta$ in T to the height h_u of the lowest-height path in G_T from u to the outside watershed ζ.*

We define the *spill-elevation* S_u of a watershed $u \neq \zeta$ in T to be the weight of the lightest edge in G_T incident to u. In order to compute the height h_u of the lowest-height path of each watershed u, we introduce the *flow graph*.

Definition 2 ((Flow graph)) *The flow graph F_T of a terrain T is a directed weighted graph with a vertex for each watershed (including the outside watershed ζ) of T. F_T contains an edge from u to v with weight S_u if the weight of edge (u, v) in G_T is the spill elevation S_u of u.*

Note that each vertex u (except ζ) in F_T has at least one outgoing edge, and may have more than one if u has several incident edges in G_T with the same (lightest) weight. Intuitively, for each watershed u the flow graph F_T contains an edge to the (first) watershed water will flow into from u as the water-level is raised. The flow graph F_T can easily be computed in a single scan of the watershed graph G_T. Before describing an algorithm for flooding T using F_T, we prove a few simple results about the structure of F_T.

Lemma 1 *If the flow graph F_T of a terrain T is acyclic then there is a directed path from each vertex in F_T to ζ.*

Proof 1 *We first prove by induction that a directed graph where each vertex has at least one outgoing edge contains a cycle; this is obviously true for a graph with 2 vertices. Assume it is true for any graph of $n \geq 2$ vertices and consider a graph of $n + 1$ vertices. The graph must contain three distinct vertices u_1, u_2, u_3 such that $u_1 \rightarrow u_2 \rightarrow u_3$ is a path (otherwise it contains a cycle). Consider the graph obtained by contracting the edge $u_1 \rightarrow u_2$. This graph has n vertices and each vertex has at least one outgoing edge, so by the induction hypothesis it contains a cycle. Therefore the uncontracted graph must also contain a cycle.*

We can now prove the lemma by contradiction; assume that there is at least one vertex u in F_T that does not have a path to the outside watershed ζ. Let X be the set of vertices in F_T that do not have a path to ζ. Since each vertex in F_T (except ζ) has an outgoing edge, u has an outgoing edge (u, v). Vertex v cannot have a path to ζ, so $v \in X$. Thus X contains at least two vertices. Since all vertices in X have at least one outgoing edge, X must contain a cycle. This contradicts the assumption that F_T is acyclic.

Lemma 2 *The weights of the edges along a directed path in F_T form a non-increasing sequence. The weights of the edges along a directed cycle in F_T are equal, and all other edges incident to the cycle have weights larger than or equal to the weight of the cycle.*

Proof 2 *Consider a path $u_1 \to u_2 \to u_3 \to \ldots u_k$ in F_T. By definition, the weight $w_{u_1u_2}$ is the spill-elevation S_{u_1} of watershed u_1 and $w_{u_2u_3}$ is the spill-elevation S_{u_2} of u_2, that is, $w_{u_1u_2} = min\{w_{u_1v} | (u_1, v) \in G_T\}$ and $w_{u_2u_3} = min\{w_{u_2v} | (u_2, v) \in G_T\}$. Since G_T must contain an edge (u_2, u_1) with weight equal to $w_{u_1u_2}$, it follows that $w_{u_2u_3} \leq w_{u_1u_2}$. Similarly we can prove that $w_{u_iu_{i+1}} \leq w_{u_{i-1}u_i}$ for any $i \in \{2, ..., k-1\}$.*

If the path is a cycle we have $u_k = u_1$ and thus $w_{u_1u_2} = w_{u_2u_3} = \ldots = w_{u_ku_1}$. By definition, (u_i, u_{i+1}) is the lightest edge in G_T incident to u_i. Thus any edge incident to u_i has at least the same weight as the weight of the cycle.

Lemma 3 *If there is a directed path from vertex u to ζ in F_T, the corresponding path exists and is the lowest-height path from u to ζ in G_T.*

Proof 3 *Let $p_1 = u \to u_1 \ldots \to \zeta$ be a path from u to ζ in F_T. By definition of F_T the corresponding (undirected) path also exists in G_T. Assume, by contradiction, that there is a lower path $p_2 = u \to v_1 \ldots \to \zeta$ from u to ζ in G_T. By Lemma 2 the maximum weight along a path in F_T is the weight of its first edge, so the height of p_1 is w_{uu_1}. The height of p_2 is at least w_{uv_1}, and by construction of F_T, $w_{uu_1} \leq w_{uv_1}$. Thus it follows that the height of p_2 is at least the same as the height of p_1, contradicting the assumption.*

1 Flooding with cycle contraction

We are now ready to discuss how to flood a terrain T, that is, how to find the height of the lowest path from each vertex in G_T to ζ. If F_T is acyclic we have found these heights: every vertex u in F_T has a path to ζ (Lemma 1), this path is the lowest path from u to ζ in G_T (Lemma 3), and the height of the path is the weight of the first edge on it (Lemma 2), i.e., the spill-elevation S_u of u. If F_T is not acyclic, we may have computed the height of the lowest-height path for some vertices in G_T (the ones with a path to ζ in F_T). The following lemma shows that the remaining paths can be computed using cycle contractions; a cycle-contraction is the process of replacing a cycle $u_1 \to u_2 ... \to u_k = u_1$ in a graph with a vertex u, and replacing all edges (u_i, v) and (v, u_i) with edges (u, v) and (v, u), respectively.

Lemma 4 *The height of the lowest-height path from any vertex u to ζ in G_T is invariant under contraction of cycles present in both G_T and F_T.*

Proof 4 *Let u be an arbitrary vertex in G_T and p the lowest-height path from u to ζ. Consider contracting a cycle C in F_T (and thus also in G_T). If C and p are disjoint, the path is obviously not affected by the contraction. Consider the case where C and p are not disjoint; since ζ is not on C, p must contain an edge leaving the cycle. Moreover, since C is in F_T the edges incident to C all have at least the same weight as the edges on C (Lemma 2). Therefore contraction of C does not change the height of the lowest-height path from u to ζ.*

It follows from Lemma 4 (and Lemma 1 through 3) that we can flood a terrain T by repeatedly finding a cycle in F_T, contracting the corresponding cycle in G_T, and recomputing/updating F_T (contracting the cycle and computing the new outgoing edge(s) of the contracted vertex using G_T). When F_T becomes acyclic all we then need to do to finish the computation is to raise each watershed u in G_T (and all watersheds merged into u by cycle contractions) to the spill-elevation S_u of u, that is, the height of u's outgoing edge in F_T. The algorithm is sketched in Figure 1. A similar approach has been employed in a number of algorithms [**?**, **?**].

1. **Initialize:** Compute G_T and F_T from terrain T.
2. **Contract:** While F_T is not acyclic do
   - Find a cycle C in F_T.
   - Contract cycle corresponding to C in G_T and compute the new spill-elevation edge of the contracted vertex.
   - Contract C in F_T and insert the new spill-elevation edge in F_T.
3. **Find raise elevations:** Raise each watershed u to S_u, that is, to the weight of the lightest edge incident to u in the final contracted graph G'_T.

Figure 1: Flooding with cycle contraction.

Intuitively, the above algorithm corresponds to repeatedly identifying two or more watersheds (a cycle in F_T) that will spill into each other when the terrain is flooded, and merge (contract) them into one watershed. The problem with this approach is that it seems difficult to predict the order in which the watersheds are merged, and therefore difficult to store F_T and G_T such that cycle detection and contraction can be performed I/O-efficiently. If we are not careful it may take $O(W)$ I/Os (and time) to identify and contract a cycle, where W is the number of watersheds in the terrain. The contracted vertex and its outgoing edge may create a new cycle which, in turn, requires $O(W)$ I/Os (and time) to identify and contract. A straightforward implementation of these ideas thus leads to an algorithm having complexity $O(W^2) = O(N^2)$.

2 Flooding with plane sweeping

The main idea of the improved flooding algorithm is to merge the watersheds in an order that avoids the expensive computation of cycles and spill-elevations of the merged watersheds. Intuitively, this order corresponds to simulating a process of uniform rise of a (subsurface) water table; conceptually, the algorithm is a bottom-up sweep of the terrain with a horizontal plane, simulating how water gradually fills watersheds as it rises uniformly across the terrain. We imagine the level of water rising with the sweep plane, and when the water level in a watershed u reaches a spill-point, it causes u to merge with an adjacent watershed (Figure **??**). If water can flow from this watershed to the outside watershed ζ, the water level in u will not increase further (and we have found the lowest-height path from u to ζ). Otherwise the level keeps rising with the sweep plane.

To perform the sweep, we process the edges in the watershed graph G_T in increasing order of weight (height). We say that a watershed is *done* when we have found the lowest-height path from it to the outside watershed ζ. Initially only ζ is *done*. When processing edge (u, v) with weight w_{uv} (meaning that water can flow between watersheds u and v at height w_{uv}), we are in one of three situations:

1. Neither u nor v is *done*: We contract the edge (u, v) in G_T.

   This corresponds to merging the two watersheds u and v. Neither of them are marked done since water still cannot flow from either of them to ζ.

2. Precisely one of u and v, say v, is *done*: We mark u as *done*.

   Since water can flow from u to v at height w_{uv} and then from v to ζ (v is done), it means that water can flow from u to ζ. We will show that since edges are processed in increasing order of weight (height), this path must be the lowest-height path from u to ζ and has height w_{uv}.

3. Both u and v are *done*: We ignore the edge.

   We have already found the lowest-height path from u and v to ζ.

Below we prove (through a series of lemmas) that when we are done with the sweep, all vertices in the final contracted graph G'_T are done and, as previously, all we need to do to finish the flooding is to raise each watershed u (and all watersheds merged into u by edge contractions) to the spill-elevation S_u of u in G'_T, that is, to the weight of the minimal weight edge incident to u in G'_T. The flooding algorithm is outlined in Figure 2. Note that during the sweep all we really need to keep track of is what watersheds (vertices) have been merged together and what watersheds are *done*. Unlike in the previous algorithm, we do not need to explicitly detect cycles or find the lowest weight edge incident to a vertex after a contraction. Even though we do not explicitly construct the flow graph F_T, and even though we contract edges instead of cycles, the final result of the algorithm is intuitively the same as of the previous algorithm; by Lemma 2, all edges of a contracted cycle have the same height. Therefore they are all hit by the sweep plane at the same time and processed after each other, resulting in the whole cycle eventually being contracted.

1. **Initialize:** Mark ζ as *done* and all other vertices as not *done*.
2. **Sweep:** Construct a list with all edges in the watershed graph G_T sorted by weight (elevation). Scan through this list and for edge (u, v) do:
    (a) If neither of u and v are *done* then contract edge (u, v).
    (b) If precisely one of u and v is *done* then mark the other one as *done*.
    (c) If u and v are both *done* then (ignore this edge and) continue with the next edge.
3. **Find raise elevation:** Raise each watershed u in the contracted watershed graph G'_T to S_u.

Figure 2: Outline of the flooding algorithm.

Lemma 5 *If a watershed u in T has a path p to ζ in G_T of height h_p, then u is* done *when the sweep has reached a height $h \geq h_p$.*

Proof 5 *Let $p = (u = u_0 \to u_1 \to u_2 \ldots \to u_{k-1} \to u_k = \zeta)$ be a path from u to ζ. When the sweep plane has reached height $h > h_p$, every edge (u_i, u_{i+1}) of p has been processed. After processing edge (u_i, u_{i+1}), u_i and u_{i+1} are either merged together (case 1) or are both marked* done *(case 2 or 3). Since $u_k = \zeta$ is* done*, it follows by induction that u_i is* done *for all i.*

Lemma 6 *If u gets marked done when the sweep reaches edge (u, v) of weight w_{uv}, then the lowest-height path p from u to ζ has height $h_p = w_{uv}$.*

Proof 6 *We first prove by induction on height (weight) that when u gets marked* done *there is a path p from u to ζ of height w_{uv}. This holds initially for ζ. Assume that it holds for any height lower than w_{uv}. If u gets marked* done *when the sweep reaches edge (u, v), then v must already be marked* done*. It must have been marked* done *when the sweep plane reached some height $h' < w_{uv}$. By induction hypothesis, this means that there is a path p' from v to ζ of height h'. Thus we have identified a path from u to ζ through v of height max$\{w_{uv}, h_{p'}\}$ $= w_{uv}$.*

Now assume by contradiction that w_{uv} is not the height h_p of the lowest path p from u to ζ, i.e., that there is a lower path of height $h_p < w_{uv}$. Consider h such that $h_p \leq h < w_{uv}$. By Lemma 5, u is done *when the sweep reaches height h, contradicting that u gets marked* done *when reaching w_{uv}.*

That the last step of the algorithm (Step 3 in Figure 2) correctly floods the terrain T now follows almost immediately.

Lemma 7 *The height of the minimal-height path from a watershed u to ζ in G_T is equal to the minimal weight edge incident to u (or the vertex representing the watershed u has been merged into) in the contracted watershed graph G'_T.*

Proof 7 *Assume that u was marked* done *when the sweep reached edge (u,v). By Lemma 6, w_{uv} is the height h_p of the lowest path p from u to ζ. Since (u,v) is not contracted it must exist in G'_T. Assume now that there exists a smaller weight edge (u,w) incident to u in G'_T. Since (u,w) was not contracted, this leads to the contradiction that u must have been marked* done *at a height $h < w_{uw}$.*

What is left to describe are the details of how we implement the algorithm efficiently. More precisely, how we implement edge contraction. The natural way to contract an edge (u,v) is to keep, say, vertex u and replace all edges (v,w) incident to v with edges (u,w) incident to u. Unfortunately, this leads to an $O(W^2)$ I/O algorithm, where W is the number of watersheds in the terrain. We improve this to $O(W)$ by not actually contracting edges but instead keeping track of what watersheds have merged. We represent the merged watersheds as the connected components of a graph C_T containing the same vertices as G_T. Initially C_T has no edges and thus initially each watershed is a separate connected component. To contract an edge (u,v) in G_T, we add this edge to C_T such that u and v are in the same connected component. Note that if u and v were already in the same connected component, the addition of the edge (u,v) does not change the connectivity of C_T (we simply add the edge without checking if u and v are in the same component since such a check could require several I/Os). When a watershed u in G_T is marked *done*, we traverse the connected component in C_T containing u and mark all the vertices (watersheds) *done*. Since we have computed the height of the minimal-height path to ζ for vertices being marked *done* (Lemma 6), we also store this height with each such vertex. After the vertices in a component are marked *done* they are never traversed again. Since a component can be traversed using a in time proportional with its size and since the size of C_T is $O(W)$ (G_T is planar), it follows that the sweep uses $O(W)$ time in total. After the sweep, the connected components in C_T represents the vertices in G'_T. The detailed algorithm is given in Figure 3.

Theorem 1 *Given the watershed graph G_T of a terrain T, flooding can be computed in $O(sort(W) + W)$ time.*

1. **Initialize:**
    - Construct C_T with a vertex for each vertex of G_T and no edges.
    - Mark ζ as *done* and all other vertices as not *done*.
2. **Sweep:** Construct a list with all the edges in the watershed graph G_T sorted by weight (height). Scan through this list and for each edge (u, v) do:
    - (a) If neither of u and v *done* then add edge (u, v) to C_T.
    - (b) If v *done* and u not *done* then
        - Compute connected component C_u containing u in C_T.
        - For every vertex $w \in C_u$, mark w as *done* and set its raise-value to w_{uv}.
    - (c) If u *done* and v not *done* then
        - Compute connected component C_v containing v in C_T.
        - For every vertex $w \in C_v$, mark w as *done* and set its raise-value to w_{uv}.
    - (d) If both and v *done* then (ignore this edge and) continue with next edge.

Figure 3: Details of the flooding algorithm.