Finding a best path for a haploid sequence

Problem: Given parameters ρ, μ and haploid (binary) sequence

$$\mathbf{w} = (w_0, w_1, \ldots w_{n-1})$$

find a best path $\mathbf{p}$ through the reference array to match $\mathbf{w}$. In our previous notation we seek to minimize

Basic strategy: Work from right to left, with branch and bound

After processing bit w_j we will have a stack of intervals A_k^j, B_k^j and an associated score S_k^j

We then carry out a pruning algorithm discarding intervals which can never yield the best path

Set $A_0^n = 0, B_0^n = m-1, S_0^n = 0$ the initial stack
For bit j Loop on bit x in ungarbled path

We set ξ the contribution to the garble (mutation) score
If $w(j) = x$, set $\xi = 0$ else set $\xi = \mu$
If $w[j]$ is missing, we set $\xi = 0$
This is a nice feature of the algorithm.
In principle missing data is handled seamlessly which yields a nice rigorous procedure for imputation.

For each interval $a = A_k^{j+1}, b = B_k^{j+1}, s = S_k^{j+1}$ we compute

$$lfx(a, b, x, \&newa, \&newb)$$

and if feasible add $(new, newb)$ to the stack for SNP j with score $s + \xi$. This deals with mutations.
Next we add a recombination state to the stack
Let s_{best} be the best score on the stack with corresponding interval a', b'. add

a state $(0, m-1)$ with score $s_{best} + \rho$
Note that this state is on the stack for SNP j *not* $j-1$

No recombination:

$$\begin{array}{ll} Stack\ (j+1) & (a,b,s) \\ & \downarrow \xi \\ Stack\ (j) & (a',b',s'=s+\xi) \\ & \downarrow \xi' \\ Stack\ (j-1) & (a'',b'',s''=s'+\xi') \end{array}$$

Recombination:

$$\begin{array}{lll} Stack\ (j+1) & (a,b,s) & \\ & \downarrow \xi & \\ Stack\ (j) & (a_{best},b_{best},s_{best}=s+\xi) \longrightarrow & (a'=0,b'=m-1,s'=s_{best}+\rho) \\ & \downarrow \psi & \downarrow \xi' \\ Stack\ (j-1) & (a'',b'',s''=s_{best}+\psi) & (a''',b''',s'''=s'+\xi') \end{array}$$

Every state has a unique ancestor.
We just pile the states into a list, and keep a back pointer

Pruning the stack

This is the key to our view of *fastLS*
Lunter has explicit thresholds but our thresholding is implicit coming from the pruning.

We create a tree whose nodes correspond to stack elements of the tree.
Let $A = (a.b, s)B = (a', b', s')$. We consider A an ancestor of B if

$$\begin{array}{rcl} (a, b) & \subset & (a', b') \\ s & > & s' \end{array}$$

if $a, b \subseteq (a', b')$ and $s' > s$ then we remove B from the stack
this is the key concept
Note that $R = (0, m-1, x)$ is on the stack and will be the root of the tree
Thus $A = (a, b, s)$ will be dropped if $s > x$ or $s \geq x$ and $A \neq R$,

The intervals that appear on the stack for SNP j have the property that for each pair A, B either one contains the other or they are disjoint.

Proof:
By construction an interval (a, b) on the stack j marks all the cards on deck j with bits corresponding to some pattern $(h_j, h_{j+1}, \ldots h_t)$ with t determined by a recombination in an ancestor. One interval contains another if and only if the corresponding subsequence is a subset.

After building the tree, the scores strictly decrease as we traverse down the tree from the root. We now can remove a node B from the stack if it's immediate children cover the interval of B After building the tree, the scores strictly decrease as we traverse down the tree from the root. We now can remove a node A from the stack if it's immediate children cover the interval of A For this one dimensional case this is especially simple. Let $A = A_0(a, b)$ (the score is irrelevant here) and A have children $A_1 = (a_1, b_1), A_2 = (a_2, b_2) \ldots A_t = (a_t, b_t)$ The intervals A_i, A_j are disjoint and so

$$A_0 = \cup_{i=1}^{t} A_i$$

if and only if

$$b_0 - a_0 + 1 = \sum_{i=1}^{t} (b_i - a_i + 1)$$

The 2 dimensional case is more complicated and discussed on the next lecture.
Next lecture: *fastLS* on diploids