## affine gap alignment

- How to score gaps in alignment to favor gap clusters rather than lots of little gaps?
  - until now, we scored all gaps equally
  - now, we score gaps as follows:
    - gap open penalty: $\gamma$
    - single-letter gap penalty: $\varepsilon$
    - → gap of length $n$ has penalty: $\gamma + n\varepsilon$
  - this linear gap formula can be incorporated into a dynamic programming alg. ☺
  - But what about quadratic/logarithmic gap penalty? → can no longer do dynamic programming ☹
- our objective function: MAX [$\alpha$(# of matches) + $\beta$(# of mismatches) + $\gamma$(# of gap clusters) + $\varepsilon$(# of single-letter gaps)]
- algorithm ~ we're gonna have 4 matrices: V, E, G, F
  - V: gives gapped alignment cost of prefixes length i,j
  - E: deals w/ gaps in x
  - F: deals w/ gaps in y
  - G: deals w/ matches + mismatches

initialization:

$V(0,j) = E(0,j) = -\gamma - j\varepsilon$

$V(i,0) = F(i,0) = -\gamma - i\varepsilon$

recurrence relationships:

$V(i,j) = \max[G(i,j), E(i,j), F(i,j)]$

$$G(i,j) = \begin{cases} V(i-1,j-1) + \alpha & \text{if } x_i = y_j \\ V(i-1,j-1) - \beta & \text{if } x_i \neq y_j \end{cases}$$

$E(i,j) = \text{MAX}[\underbrace{E(i,j-1)}_{\text{this considers continuing a gap}}, \underbrace{V(i,j-1) - \gamma}_{\text{this considers starting a new gap cluster}}] - \varepsilon$

$F(i,j) = \text{MAX}[F(i-1,j), V(i-1,j) - \gamma] - \varepsilon$

we shall now direct our attention to the beautiful powerpoint on the website for all the details

* it uses slightly different notation