COMP3421

Ray Tracing

Robert Clifton-Everest

Email: robertce@cse.unsw.edu.au

Recap: Global Lighting

- The lighting equation we looked at earlier only handled direct lighting from sources:

$$I = \boxed{I_a \rho_a} + \sum_{l \in \text{lights}} I_l \left(\rho_d (\hat{\mathbf{s}}_\mathbf{l} \cdot \hat{\mathbf{m}}) + \rho_{sp} \left(\hat{\mathbf{r}}_\mathbf{l} \cdot \hat{\mathbf{v}} \right)^f \right)$$

- We added an ambient fudge term to account for all other light in the scene.

- Without this term, surfaces not facing a light source are black.

Recap: Global lighting

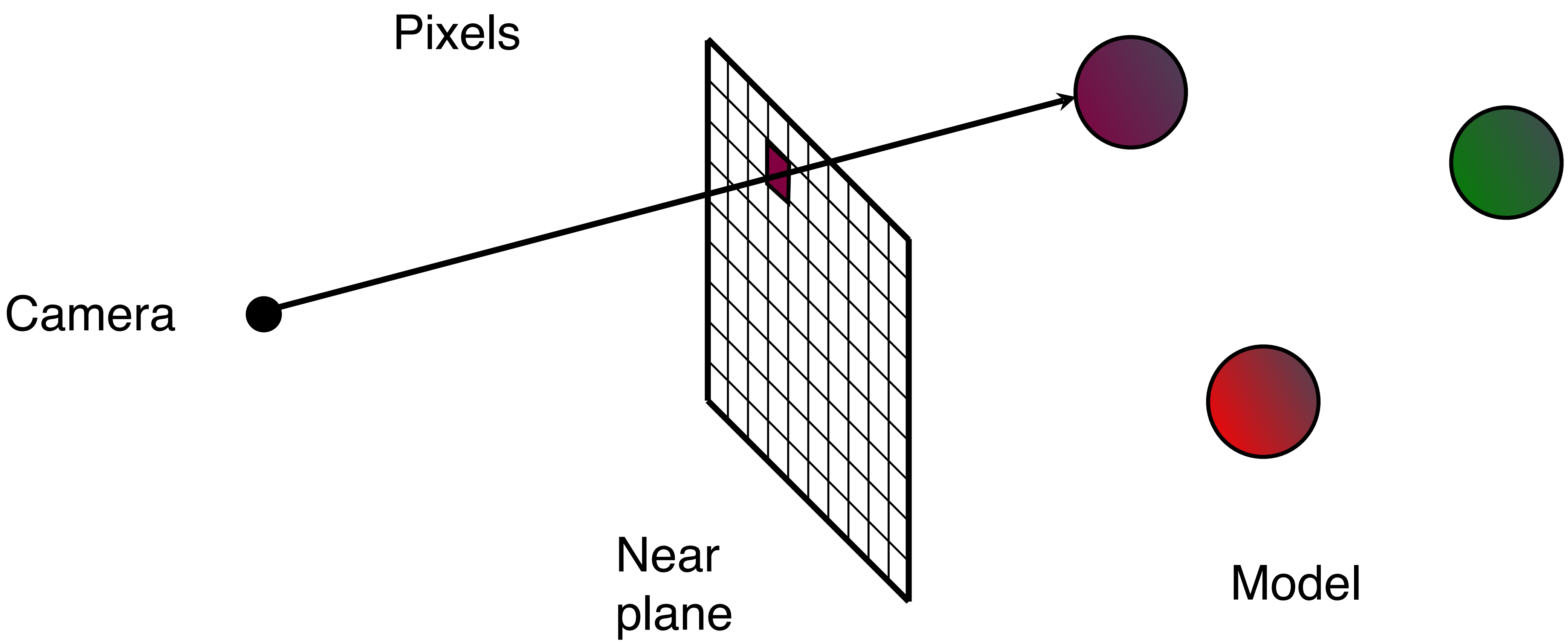
- In reality, the light falling on a surface comes from everywhere. Light from one surface is reflected onto another surface and then another, and another, and...

- Methods that take this kind of multi-bounce lighting into account are called global lighting methods.

Raytracing and Radiosity

- There are two main methods for global lighting:
  - Raytracing models specular reflection and refraction.
  - Radiosity models diffuse reflection.
- Both methods are computationally expensive and are rarely suitable for real-time rendering.

Rays



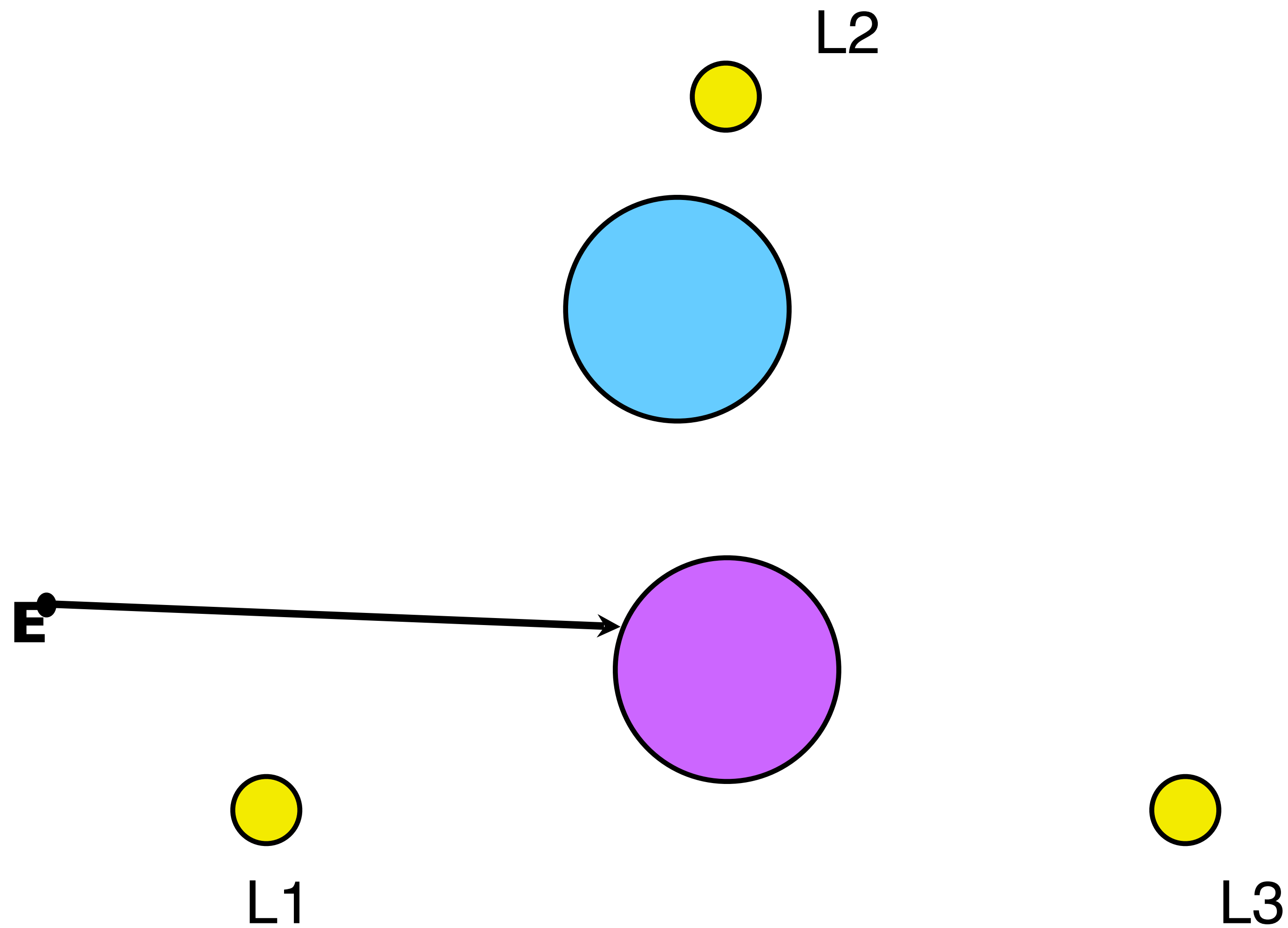
Shadows

- We can add shadows very simply.
- At each hit point we cast a new ray towards each light source. These rays are called shadow feelers.
- If a shadow feeler intersects an object before it reaches the source, then omit that source from the illumination equation for the point.

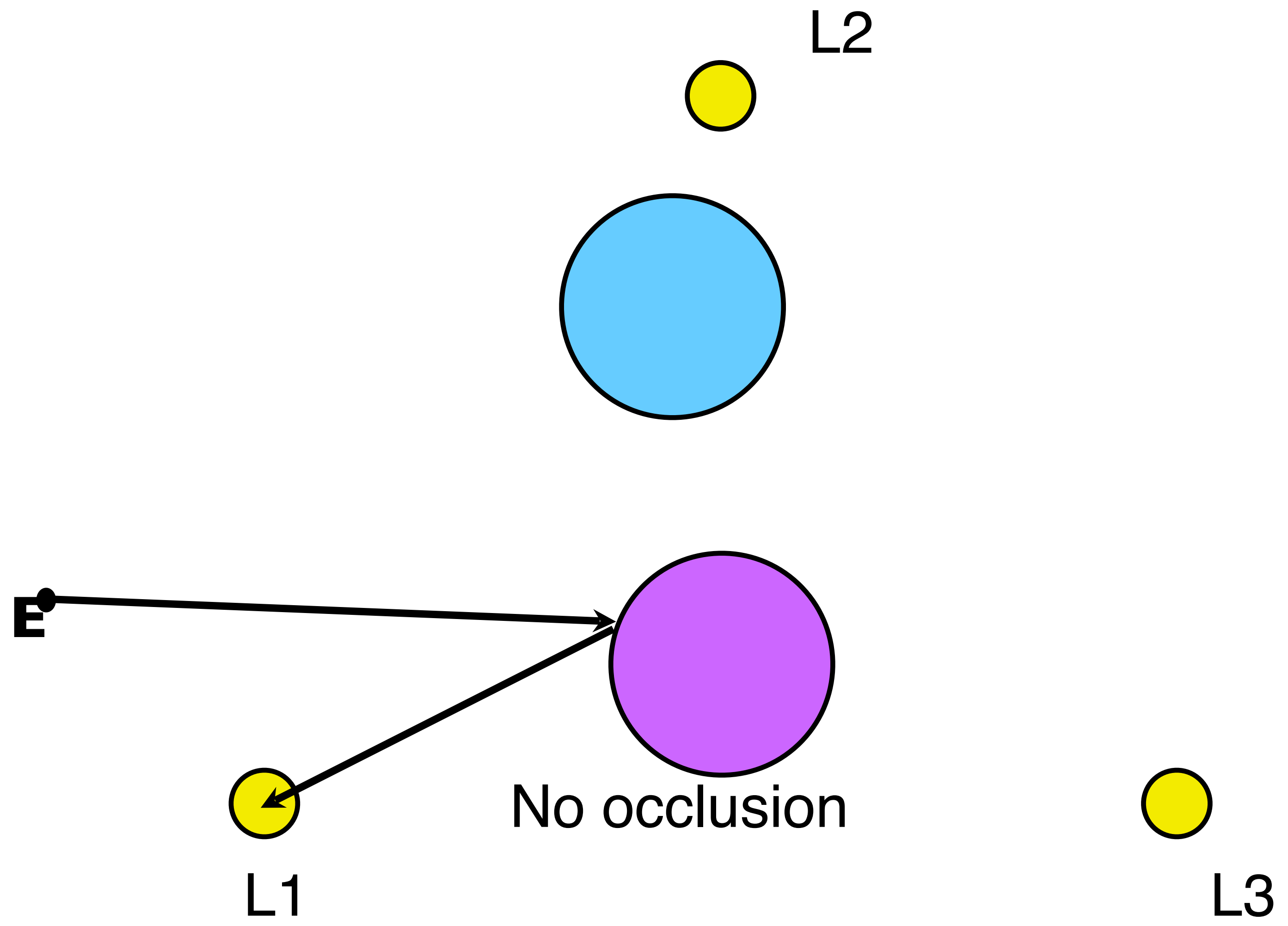
Self-shadows

- We need to take care when the shadow is cast by the hit object itself.
- The shadow feeler will always intersect the hit object at time t=0.
- This intersection is only relevant if the light is on the opposite side of the object.

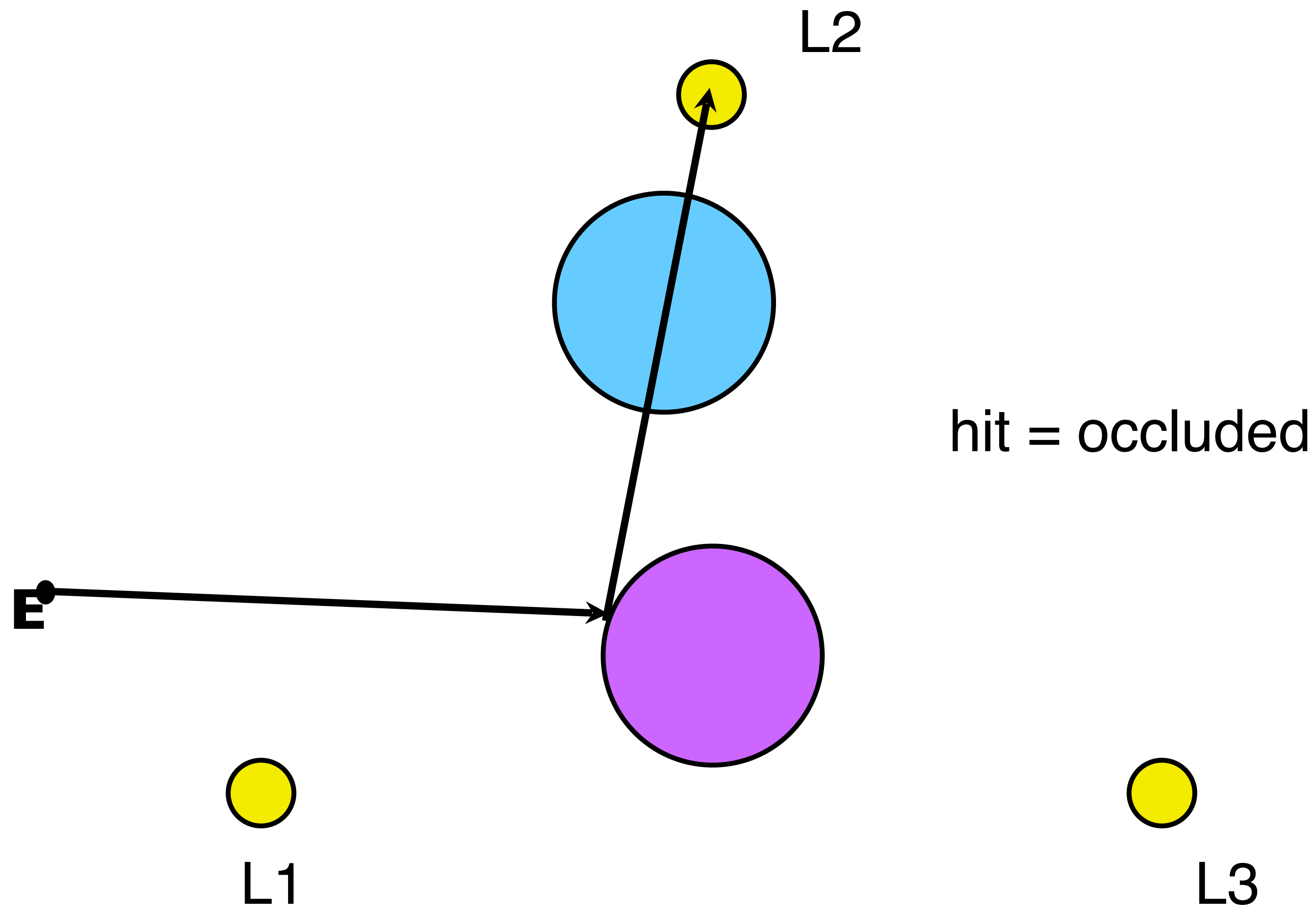
Example



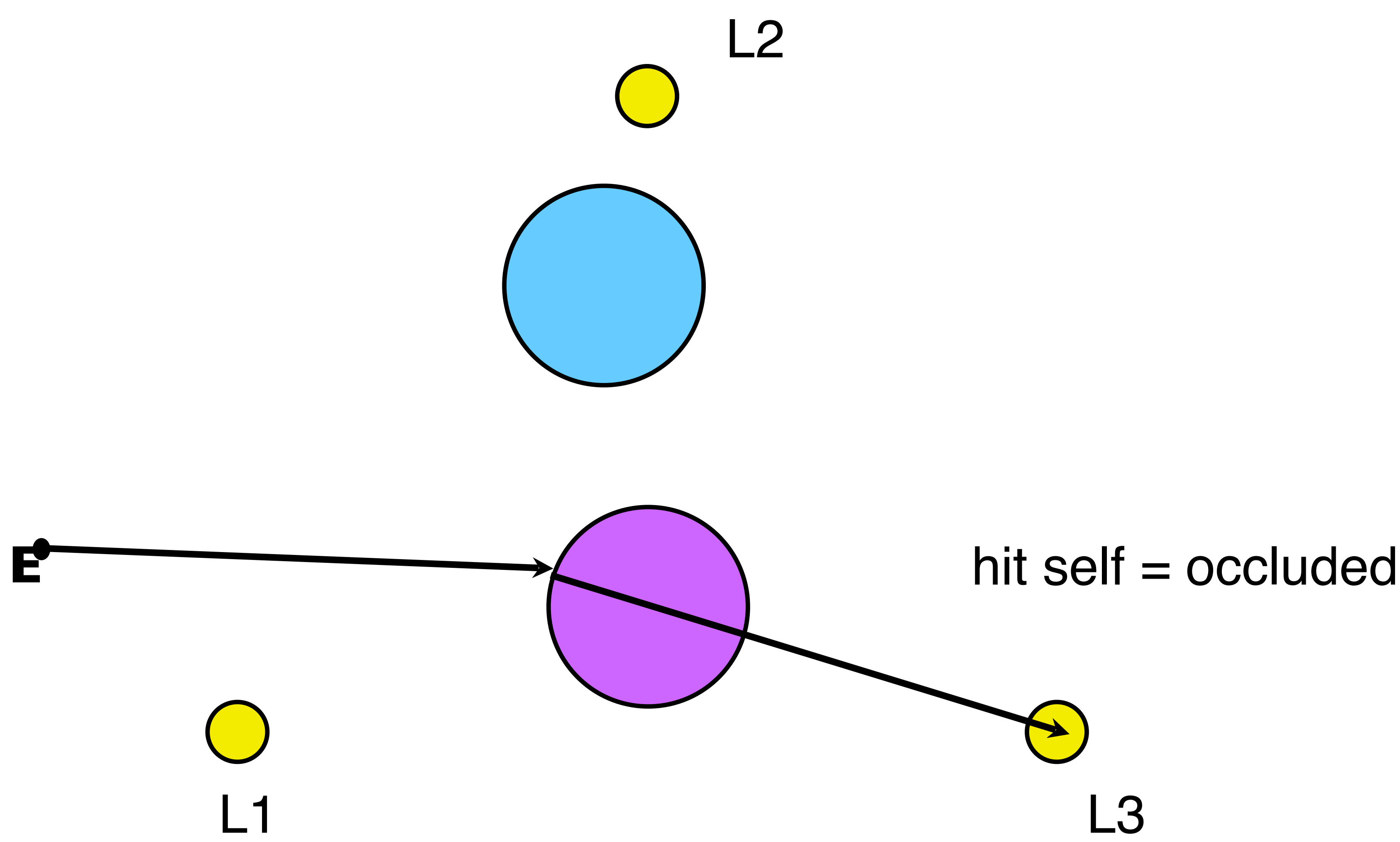
Example



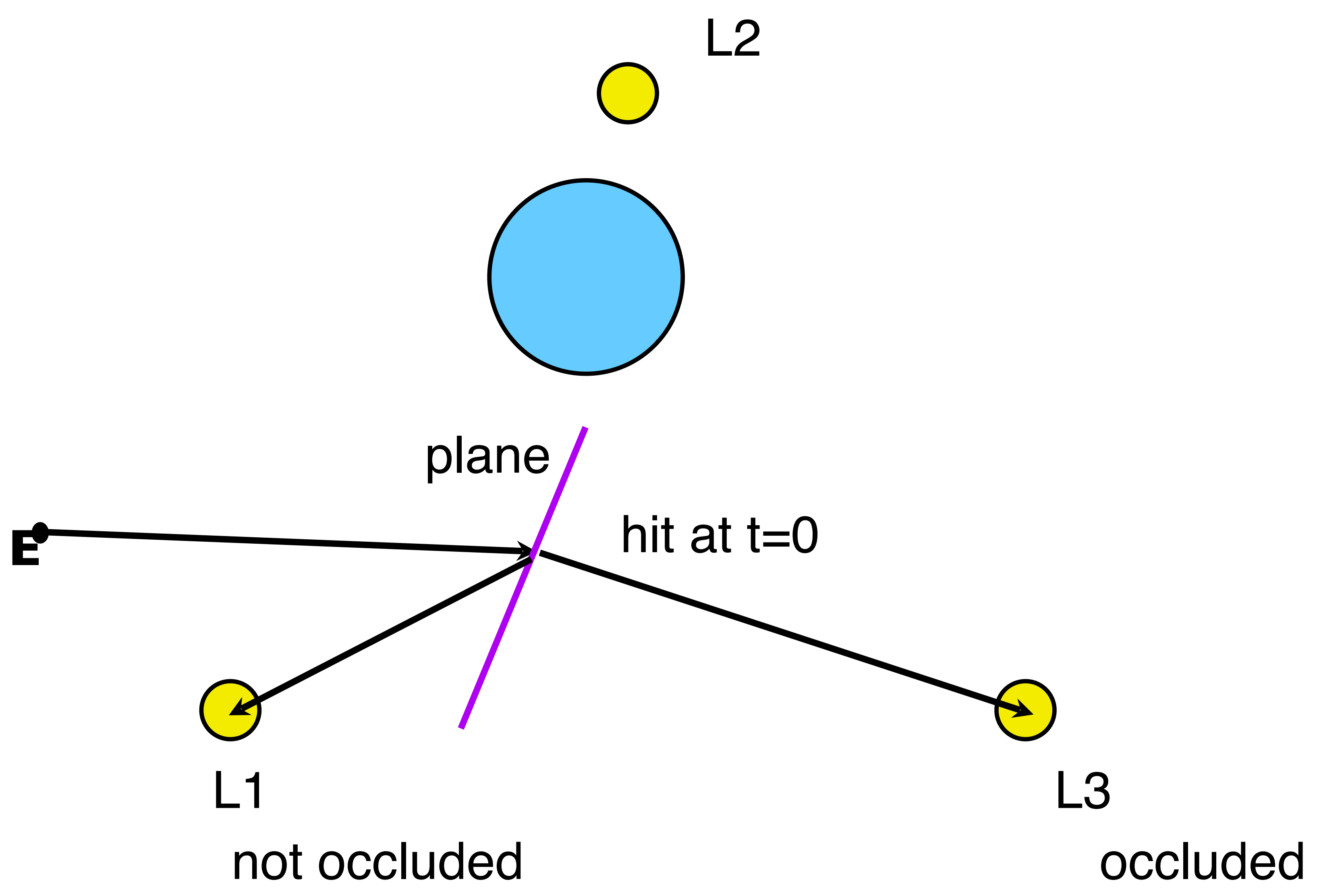
Example



Example



Example

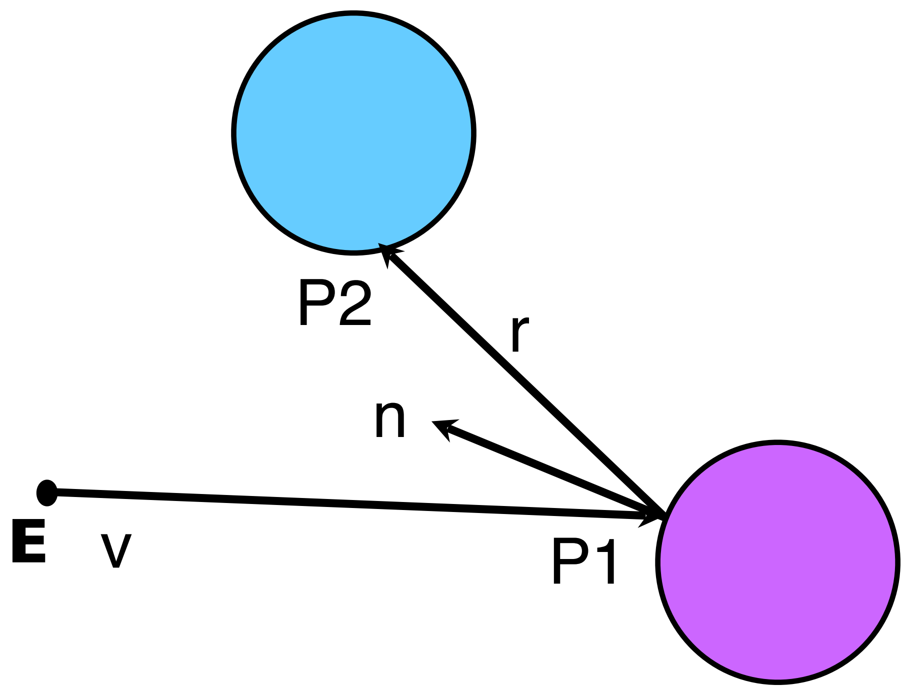


PseudoCode

```
Trace primary ray
if (hit is null)
  set (x,y) to background
else
  set (x,y) = ambient color
  Trace secondary ray to each light
    if not blocked from light
      (x,y) += contribution from light
```

Reflections

- We can now implement realistic reflections by casting further reflected rays.

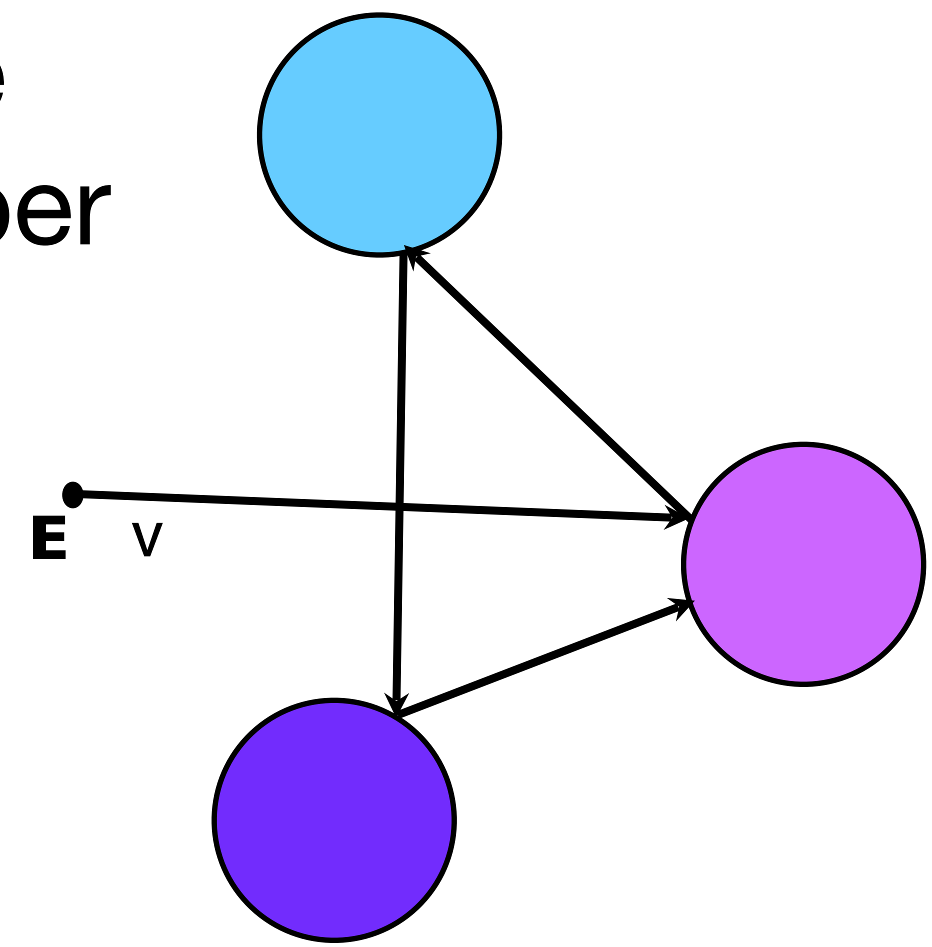


$$\mathbf{r} = \mathbf{v} - 2(\mathbf{v} \cdot \mathbf{n})\mathbf{n}$$

Reflections

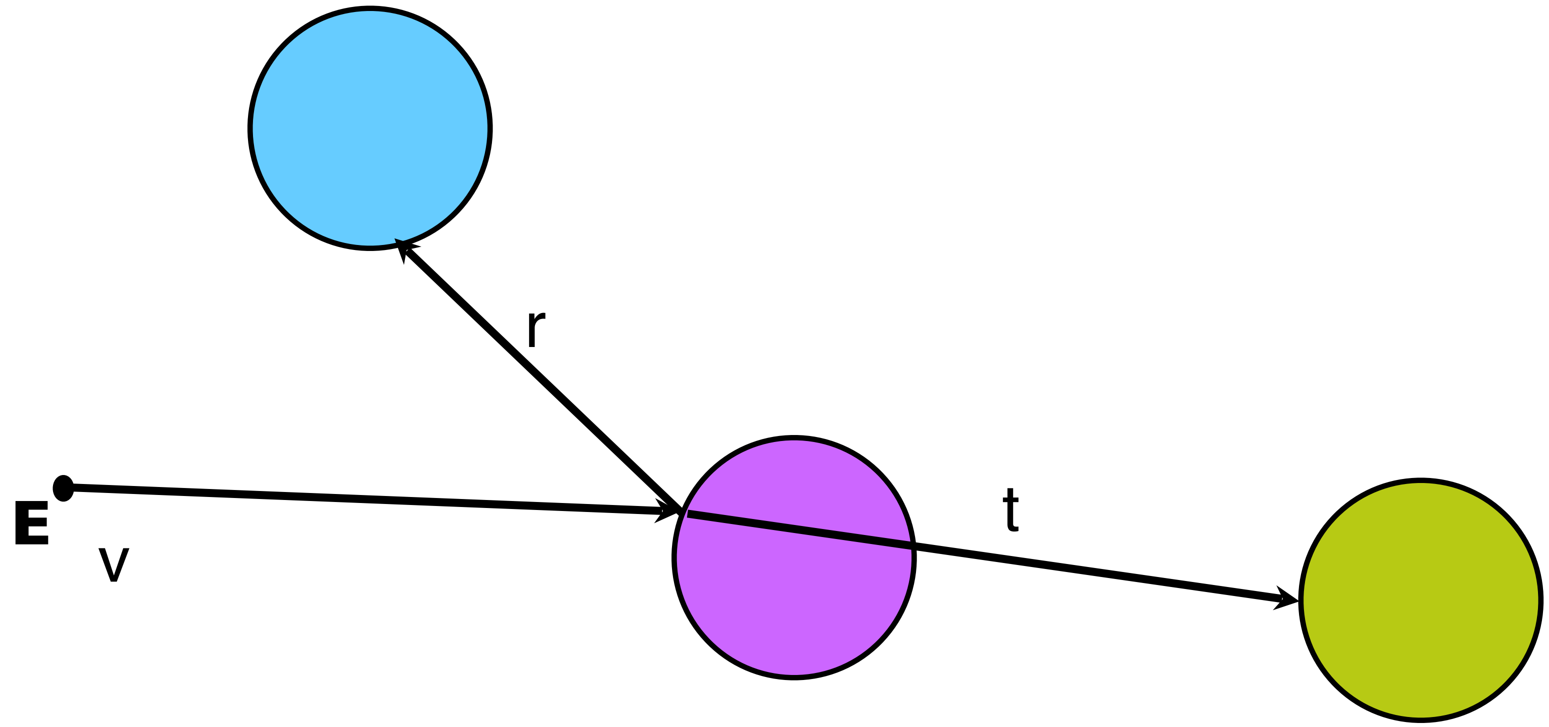
- Reflected rays can in turn be reflected off another object and another.

- We usually write our code to stop after a fixed number of reflections to avoid infinite recursion.



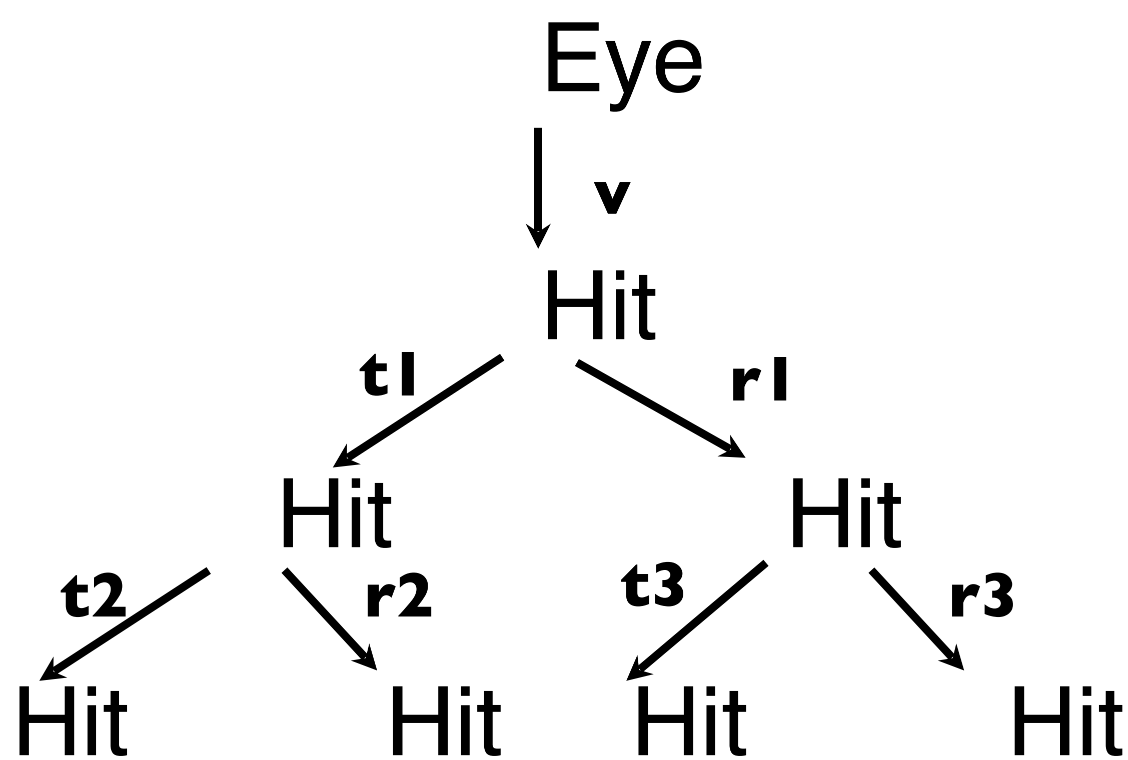
Transparency

- We can also model transparent objects by casting a second ray that continues through the object.



Transparency

- Transparency can also be applied reflexively, yielding a tree of rays.



Illumination

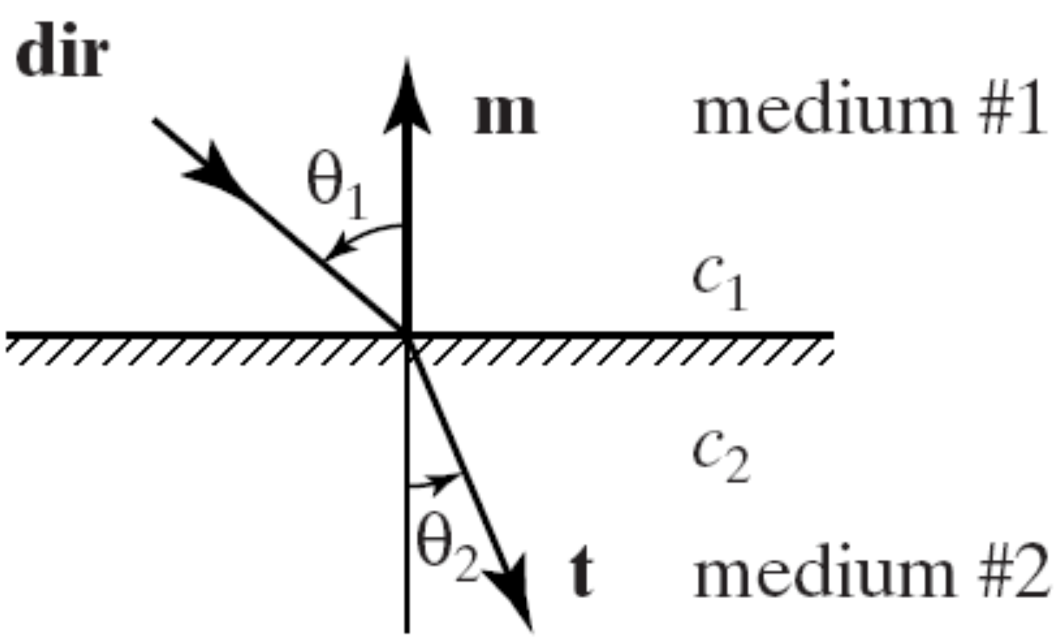
- The illumination equation is extended to include reflected and transmitted components, which are computed recursively:

$$I(P) = I_{amb} + I_{dif} + I_{spe} + \boxed{I(P_{ref}) + I(P_{tra})}$$

- We will need material coefficients to attenuate the reflected and transmitted components appropriately.

Refraction of Light

- When a light ray strikes a transparent object, a portion of the ray penetrates the object. The ray will change direction from **dir** to **t** if the speed of light is different in medium 1 and medium 2. Vector **t** lies in the same plane as **dir** and the normal **m**.



Refraction

- To handle transparency appropriately we need to take into account the refraction of light.
- Light bends as it moves from one medium to another. The change is described by Snell's Law:

$$\frac{\sin\theta_1}{c_1} = \frac{\sin\theta_2}{c_2}$$

- where c1 and c2 are the speeds of light in each medium.

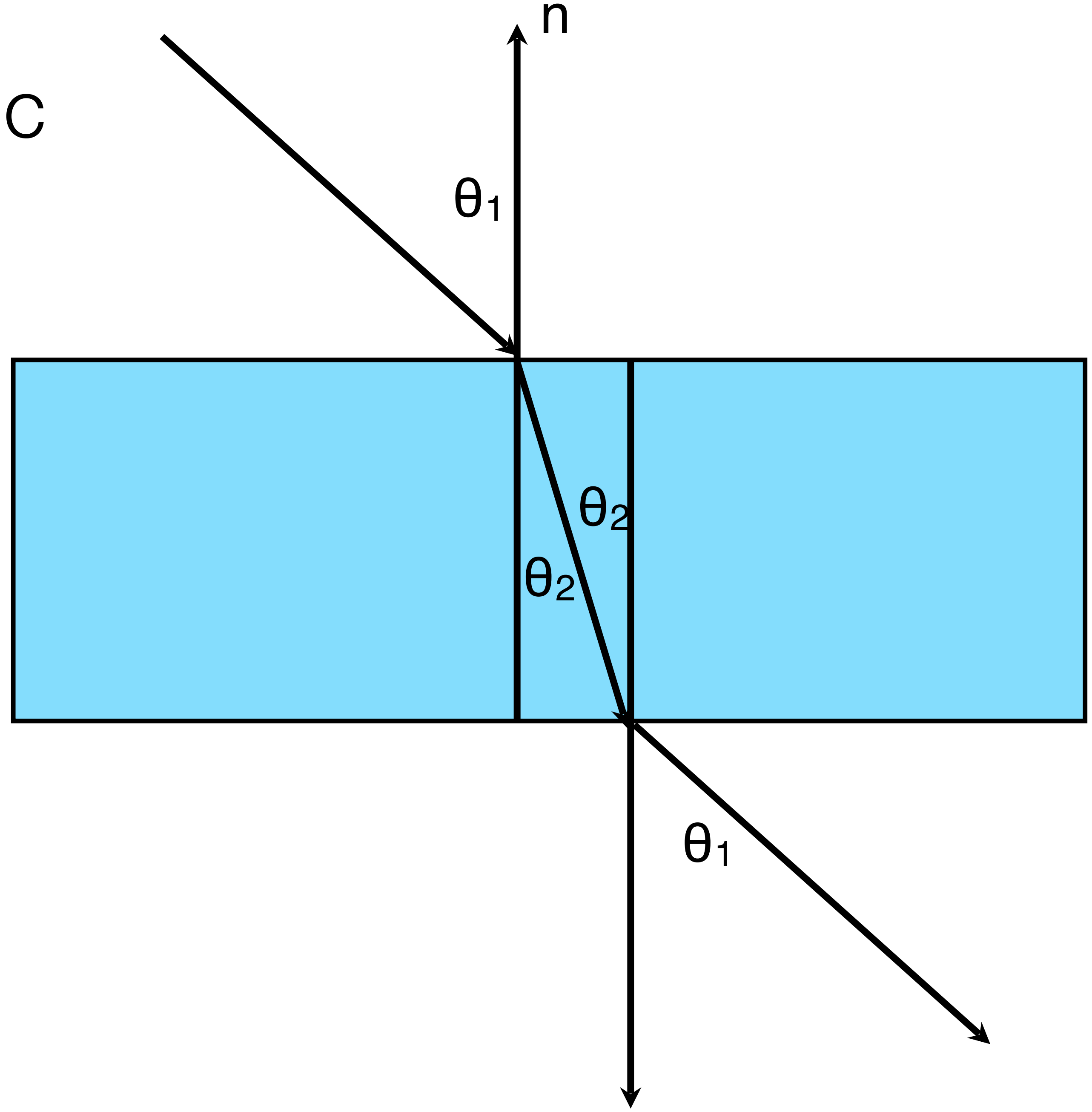
Example Snell's law

Air:

$c_1 = 99.97\%\ c$

Glass:

$c_2 \sim= 55\%\ c$



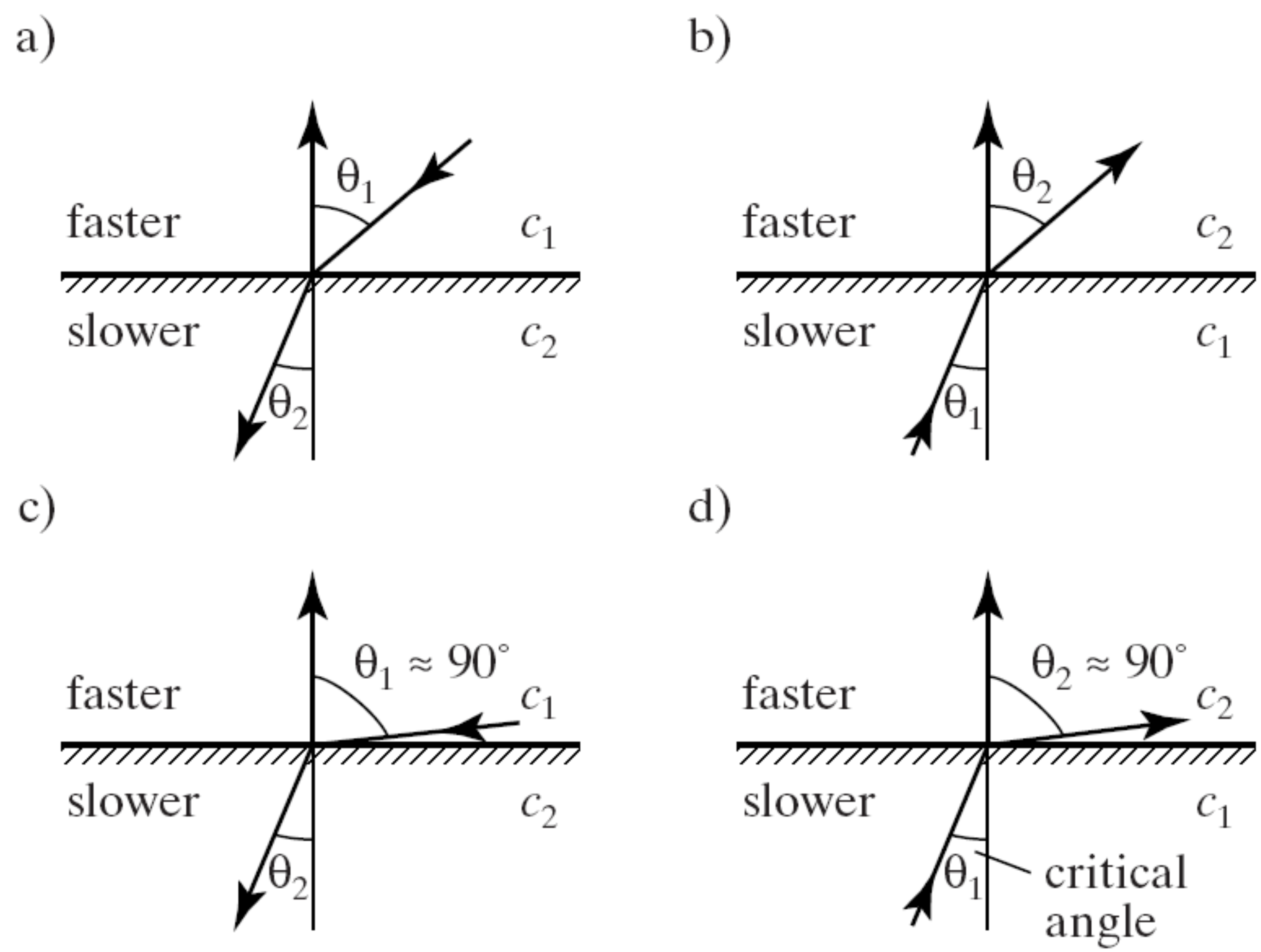
Example

- Suppose medium 2 is some form of glass in which light only travels 55% as fast as in medium 1 which is the air. Suppose the angle of incidence of the light is 60 degrees from the normal. What is the angle of the transmitted light?

Refraction

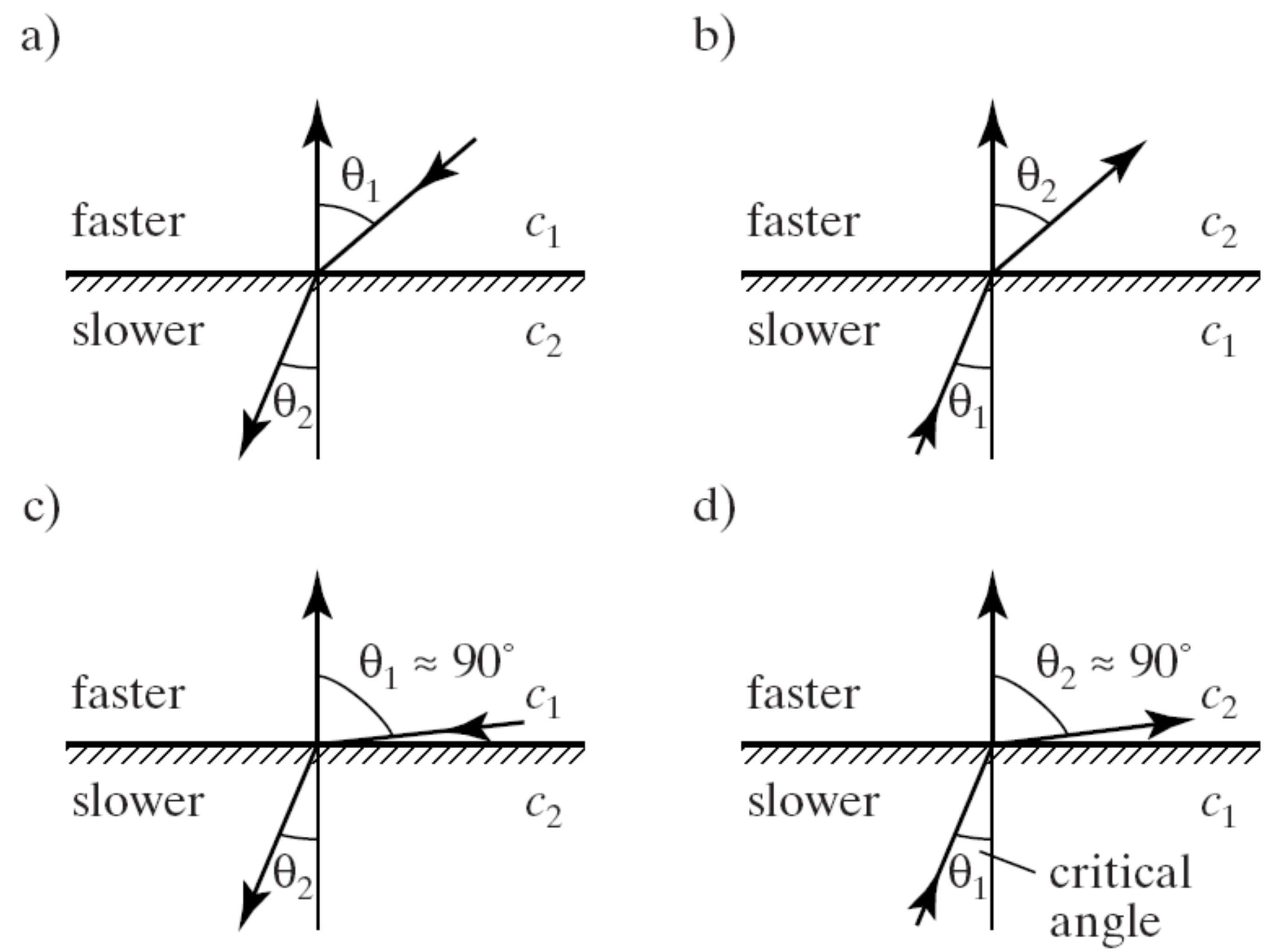
- The figure (a) shows light moving from the faster medium to the slower, and (b) shows light moving from the slower to the faster medium.

- The angles pair together in the same way in both cases; only the names change.



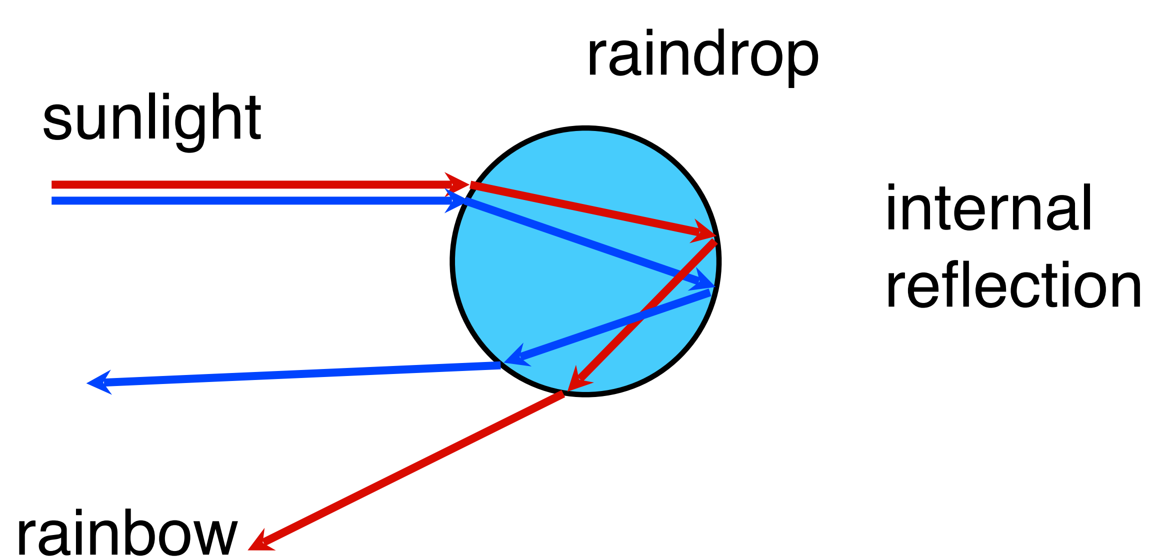
Refraction

- In (c) and (d), the larger angle has become nearly 90^0. The smaller angle is near the **critical angle**: when the smaller angle (of the slower medium) gets large enough, it forces the larger angle to 90^0. A larger value is impossible, so no light is transmitted into the second medium. This is called **total internal reflection**.



Refraction

- Different wavelengths of light move at different speeds (except in a vacuum).
- So for maximum realism, we should calculate different paths for different colours.



Refraction of Light

- Simplest to model transparent objects so that their index of refraction does not depend on wavelength.
- To do otherwise would require tracing separate rays for each of the color components, as they would refract in somewhat different directions.
- This would be expensive computationally, and would still provide only an approximation, because an accurate model of refraction should take into account a large number of colors, not just the three primaries.

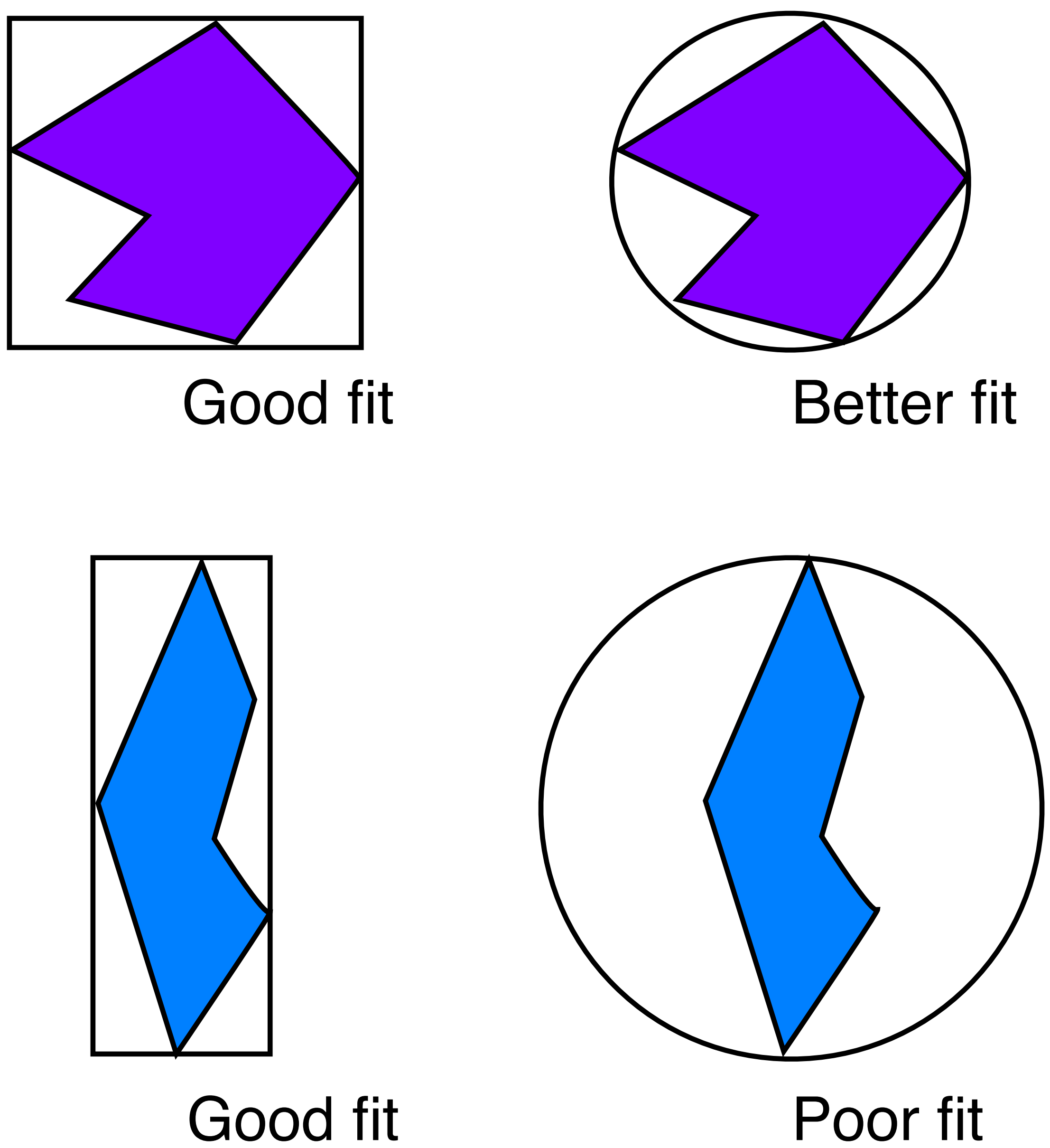
Optimisation

- Testing collisions for more complex shapes (such as meshes) can be very time consuming.
- In a large scene, most rays will not hit the object, so performing multiple expensive collision tests is wasteful.
- We want fast ways to rule out objects which will not be hit.

Extents

- Extents are bounding boxes or spheres which enclose an object.
- Testing against a box or sphere is fast.
- If this test succeeds, then we proceed to test against the object.
- We want tight fitting extents to minimise false positives.

Extents



Computing extents

- To compute a box extent for a mesh we simply take the min and max x, y and z coordinates over all the points.

- To compute a sphere extent we find the centroid of all the vertices by averaging their coordinates. This is the centre of the sphere.

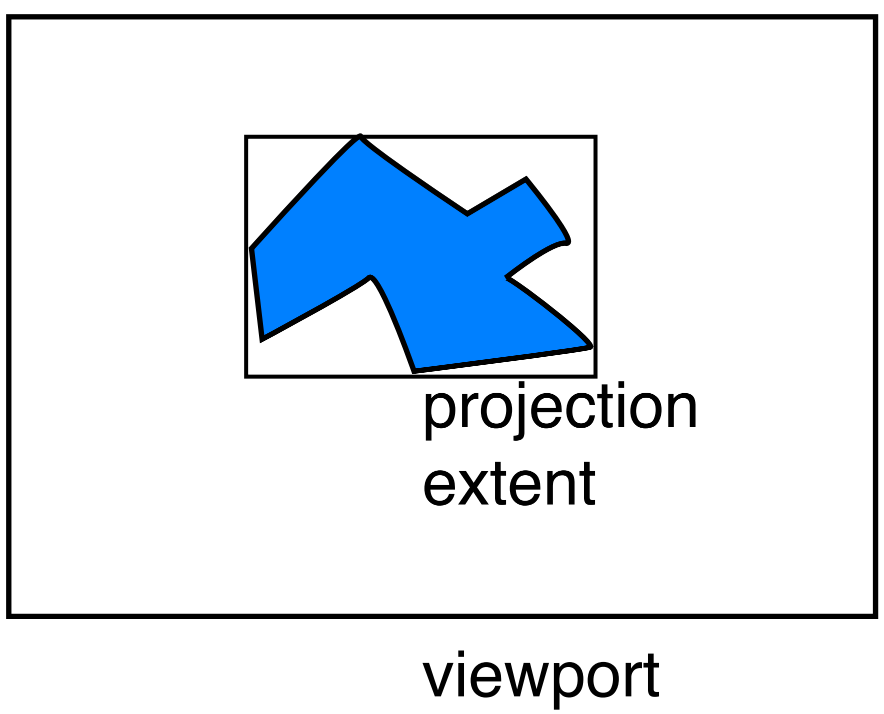
- The radius is the distance to the vertex farthest from this point.

Projection extents

- Alternatively, we can build extents in screen space rather than world space.
- A projection extent of an object is a bounding box which encloses all the pixels which would be in the image of the object (ignoring occlusions).
- Pixels outside this box can ignore the object.
- Doesn’t work for shadow feelers or reflected rays

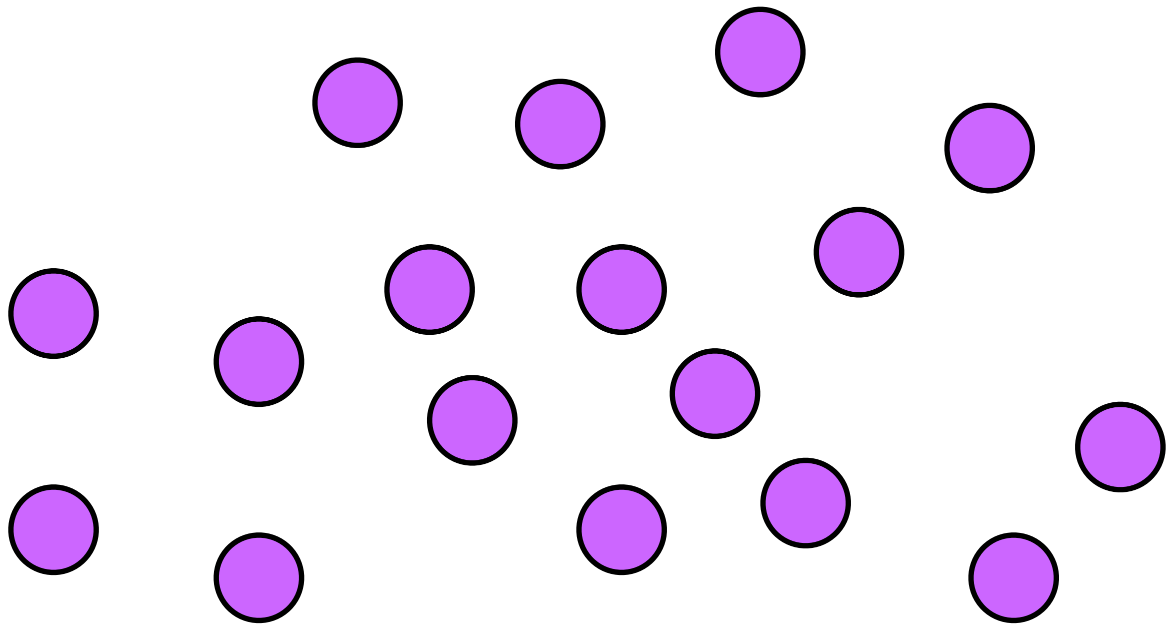
Projection extents

- We can compute a projection extent of a mesh by projecting all the vertices into screen space and finding the min and max x and y values.



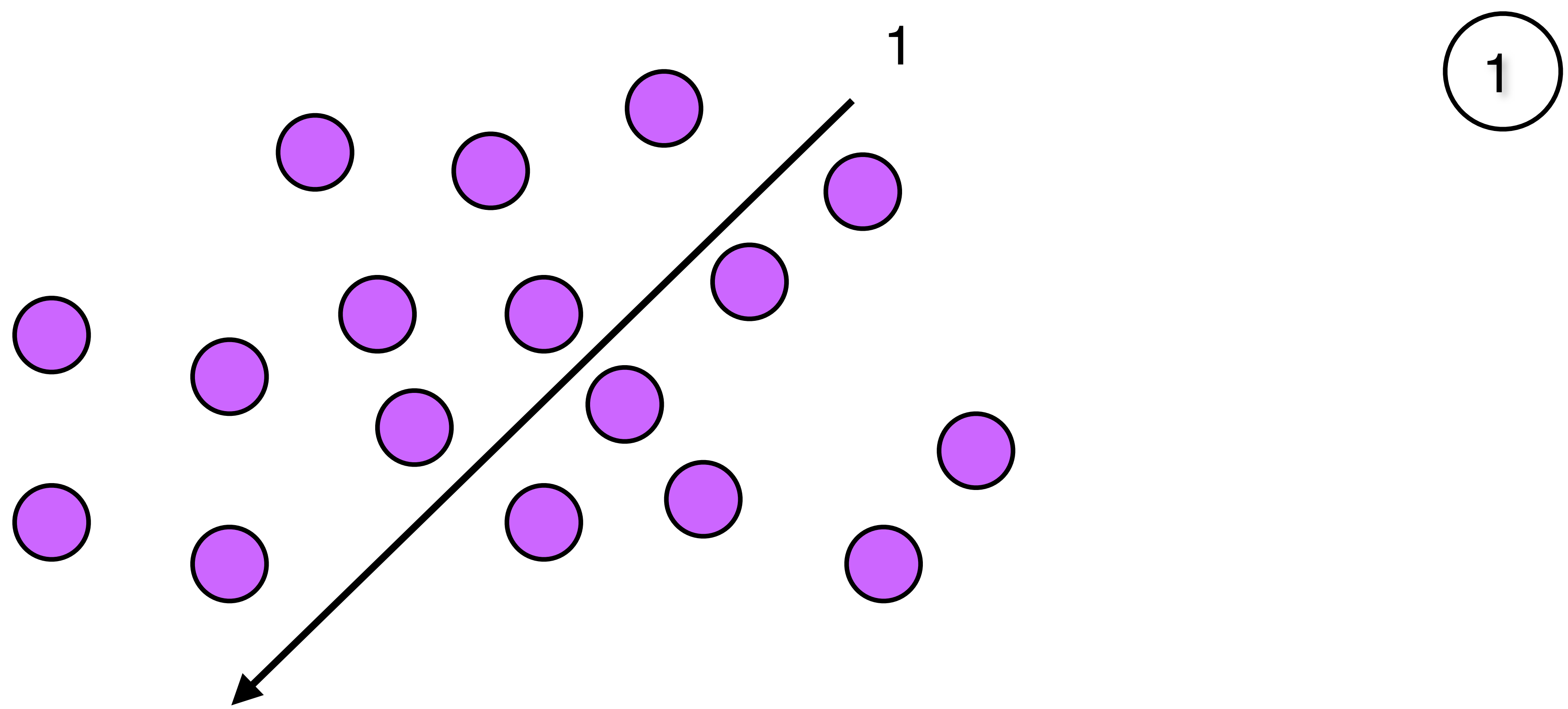
Binary Space Partitioning

- Another approach to optimisation is to build a Binary Space Partitioning (BSP) tree dividing the world into cells, where each cell contains a small number of objects.



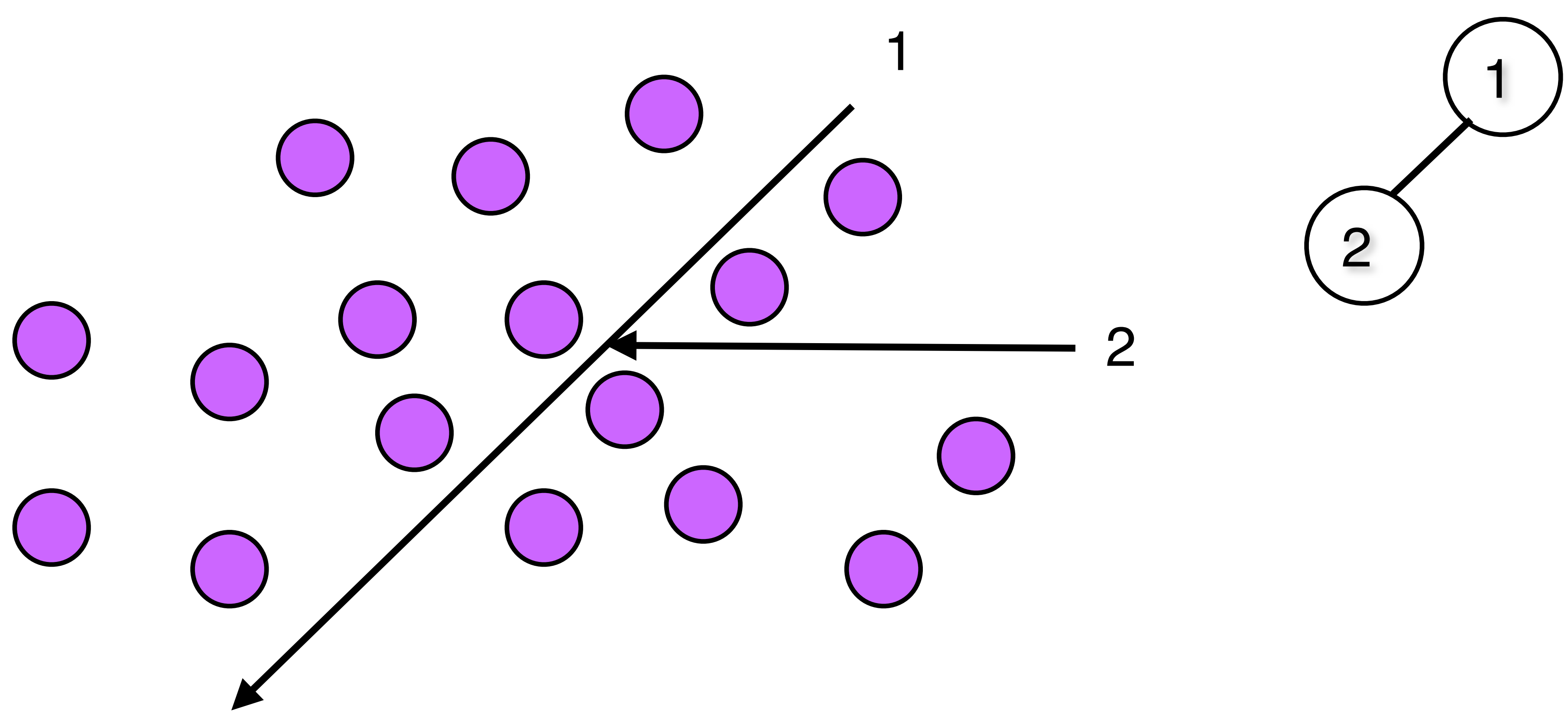
BSPs

- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



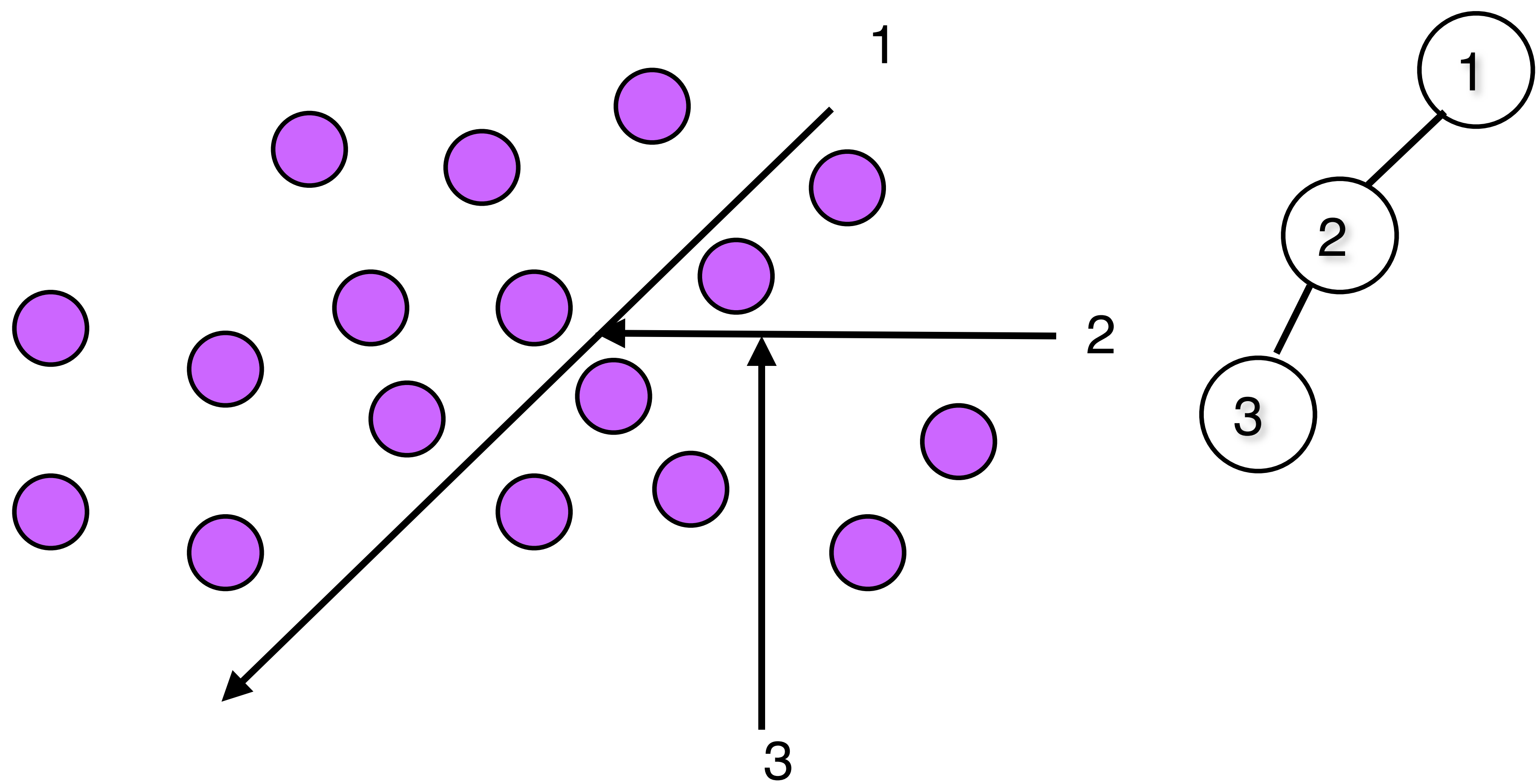
BSPs

- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



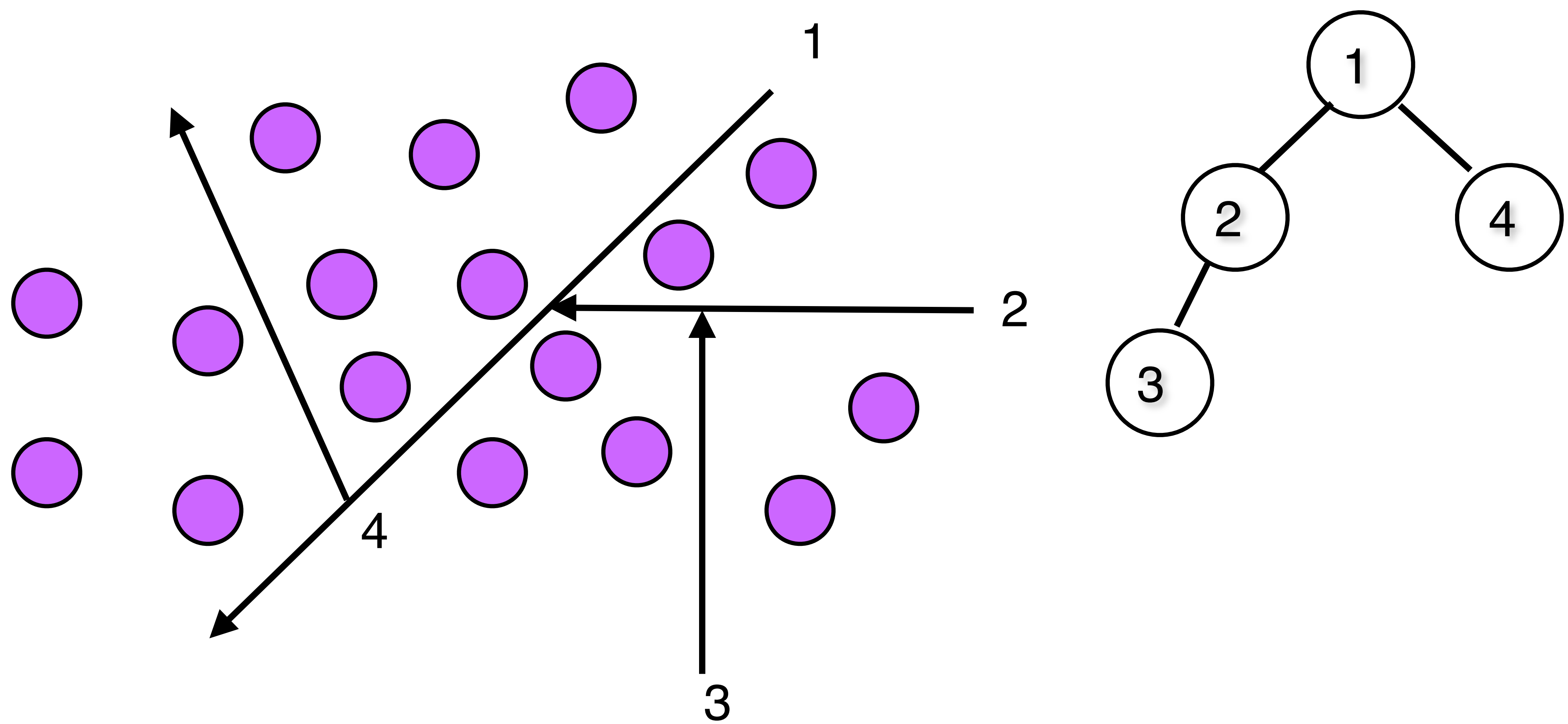
BSPs

- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



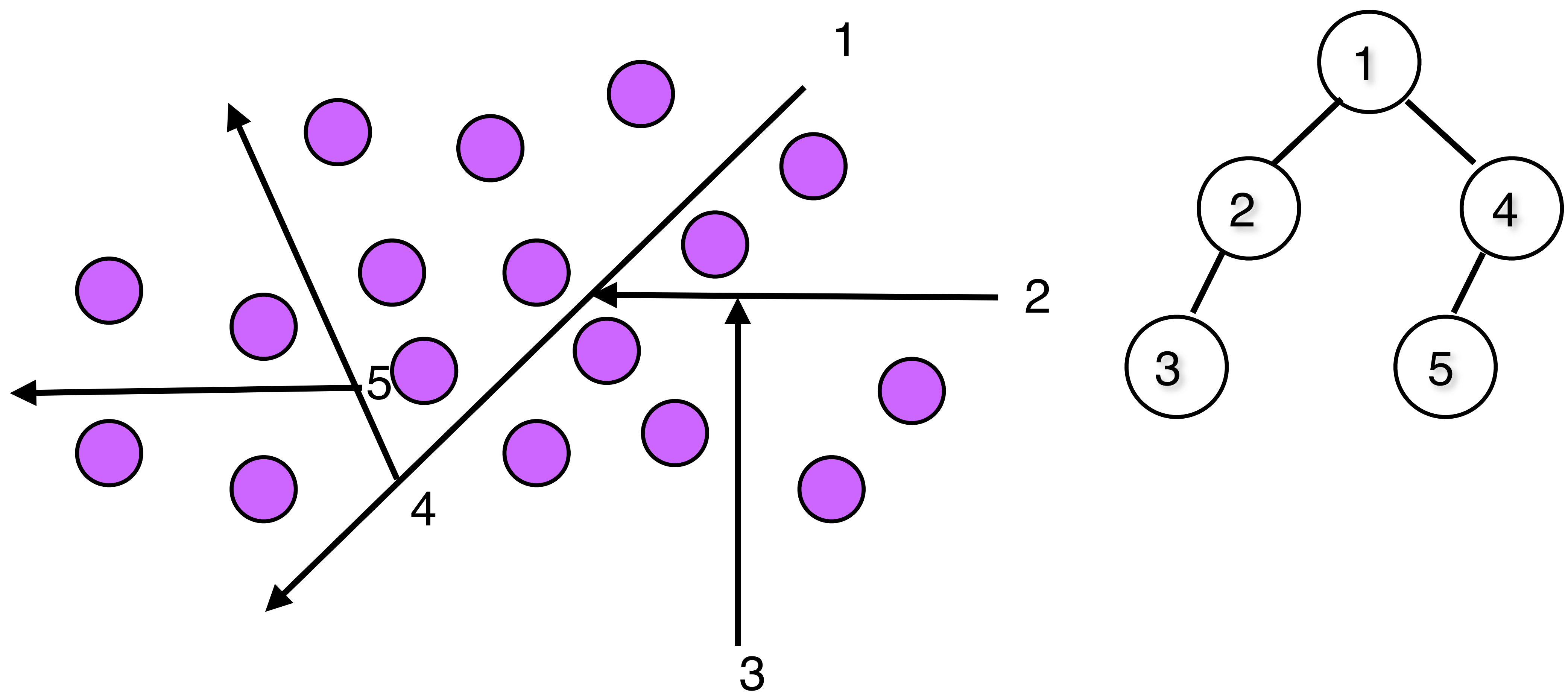
BSPs

- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



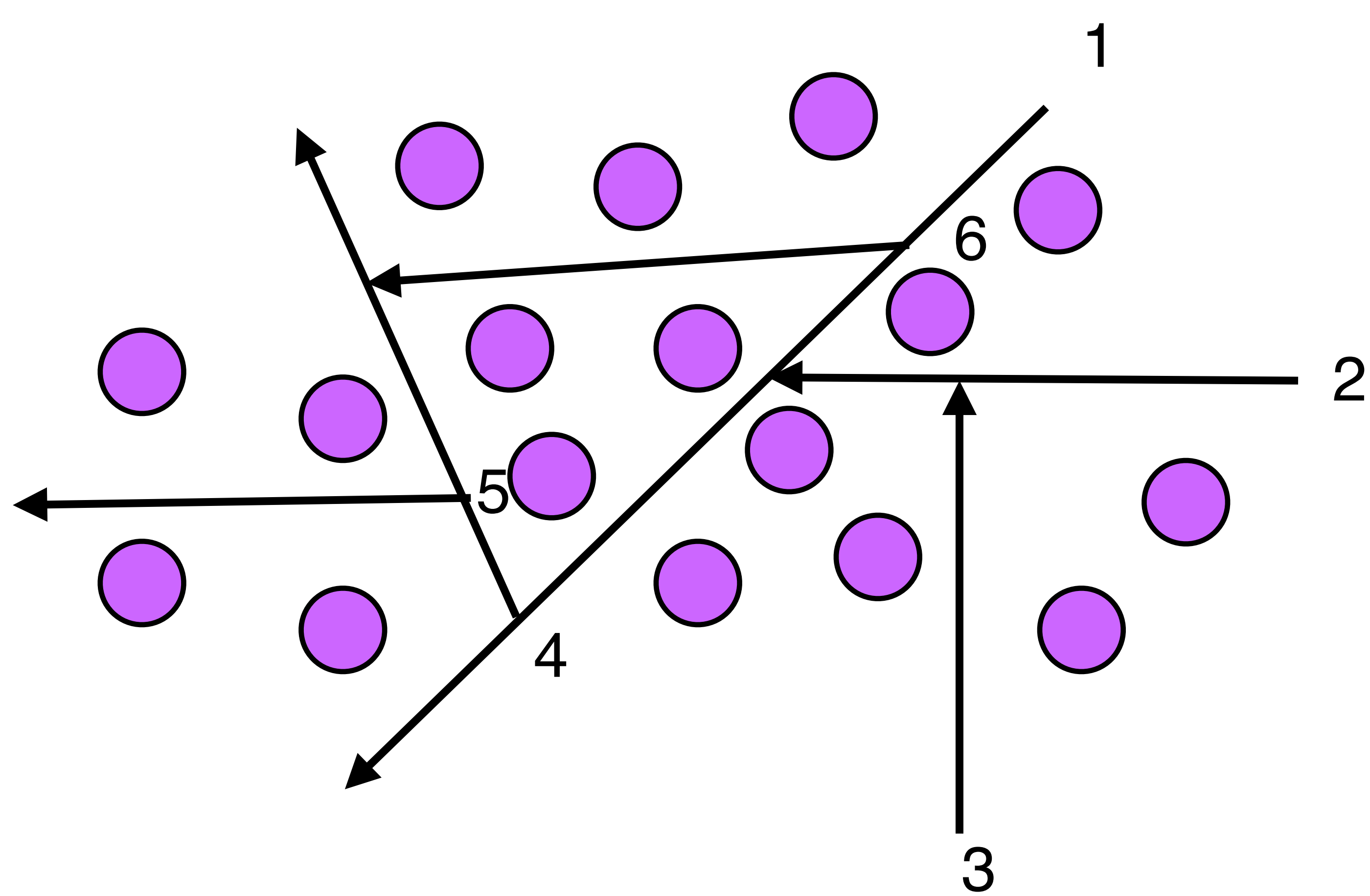
BSPs

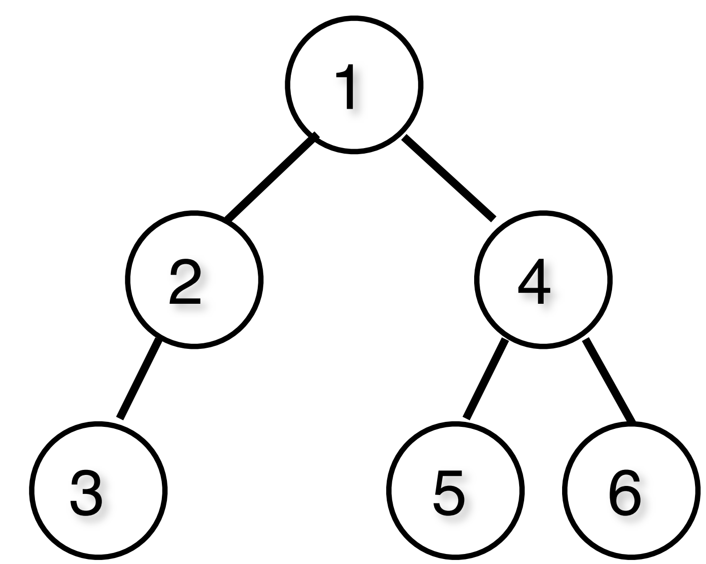
- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



BSPs

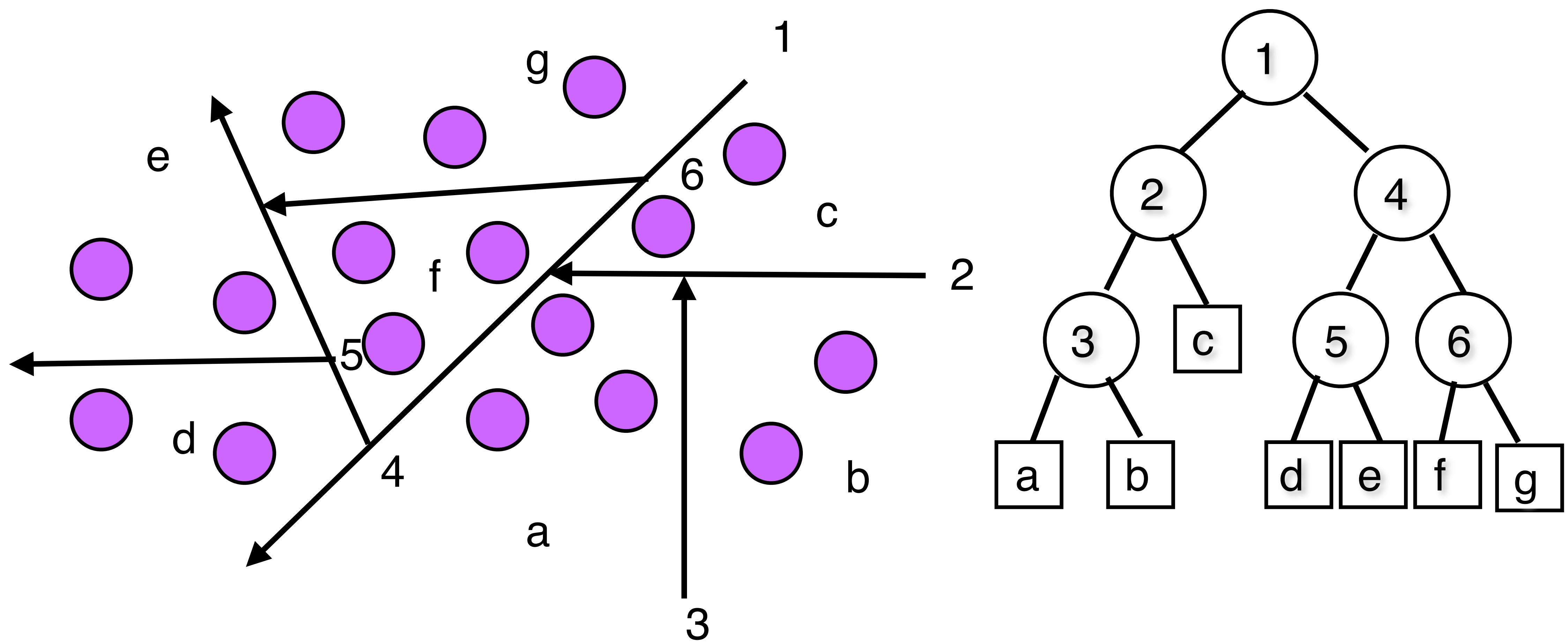
- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.





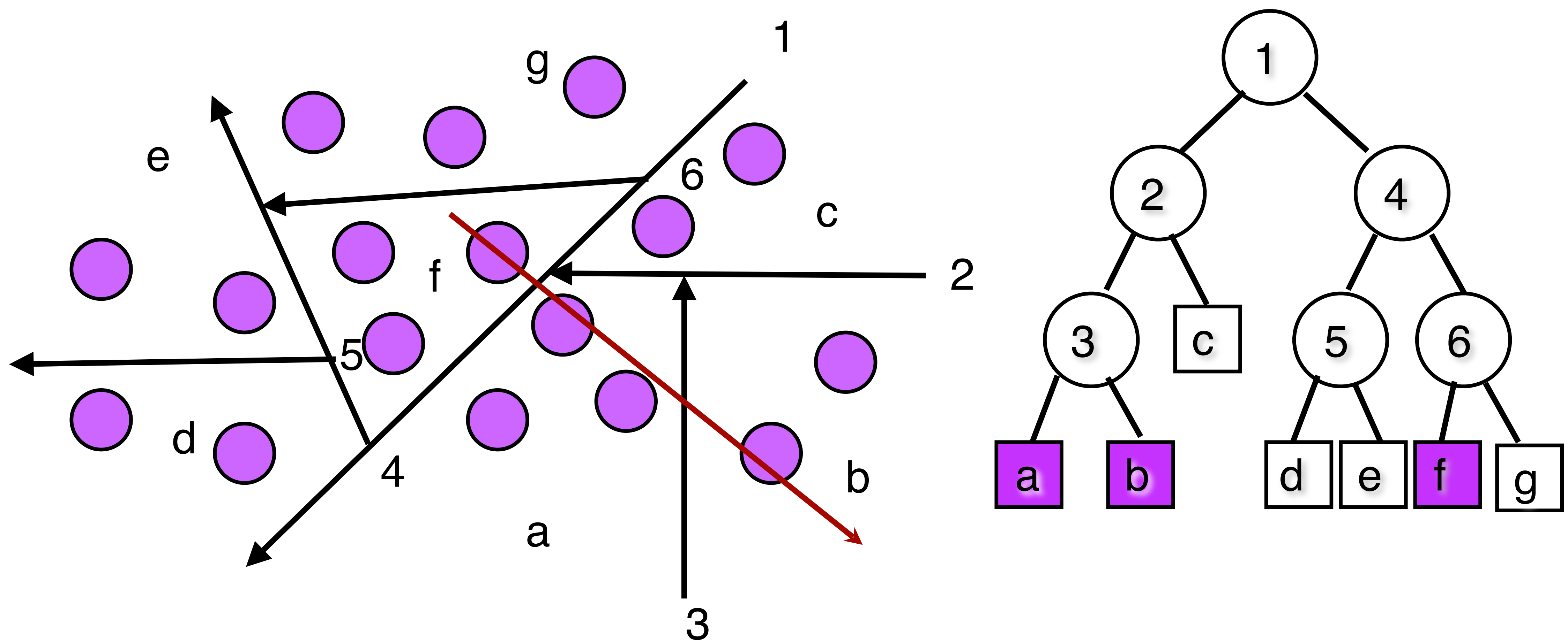
BSPs

- Another approach to optimisation is to build a BSP tree dividing the world into cells, where each cell contains a small number of objects.



Traversing the tree

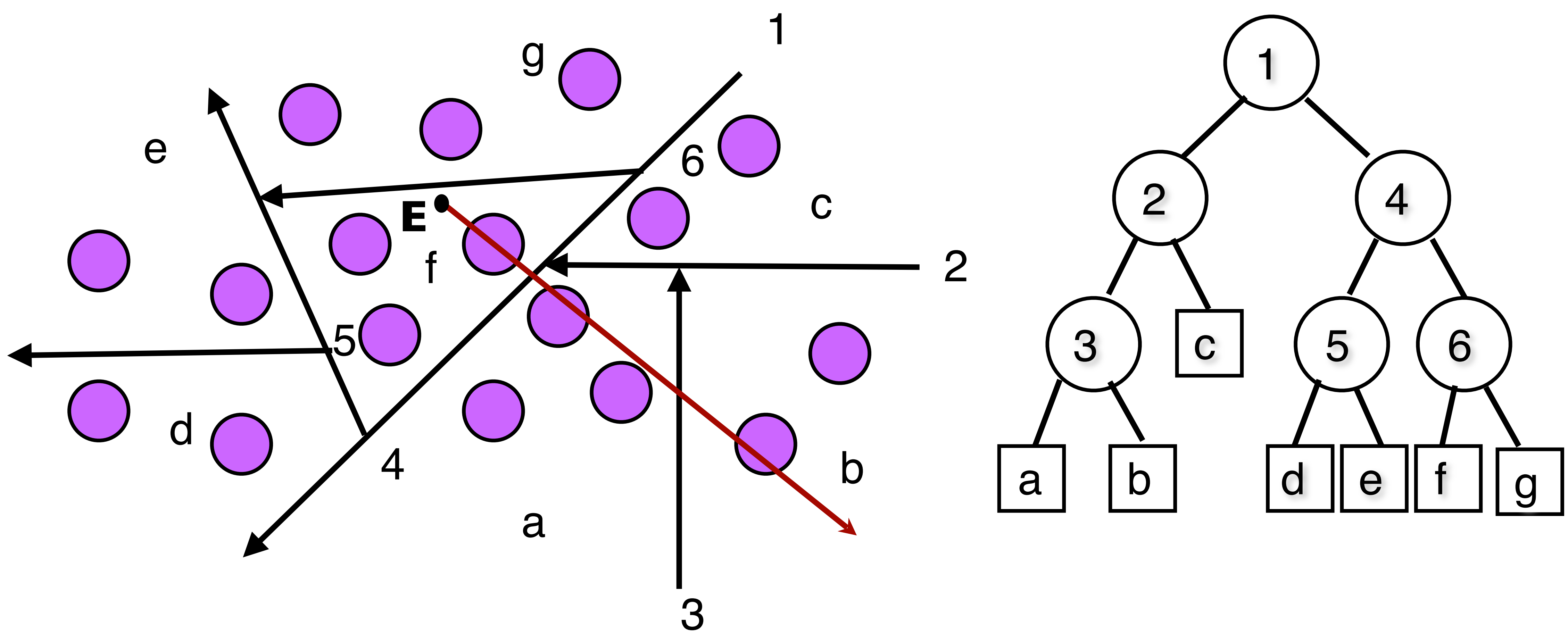
- In this case we do not want to traverse the entire tree. We only want to visit the leaves the ray passes through.



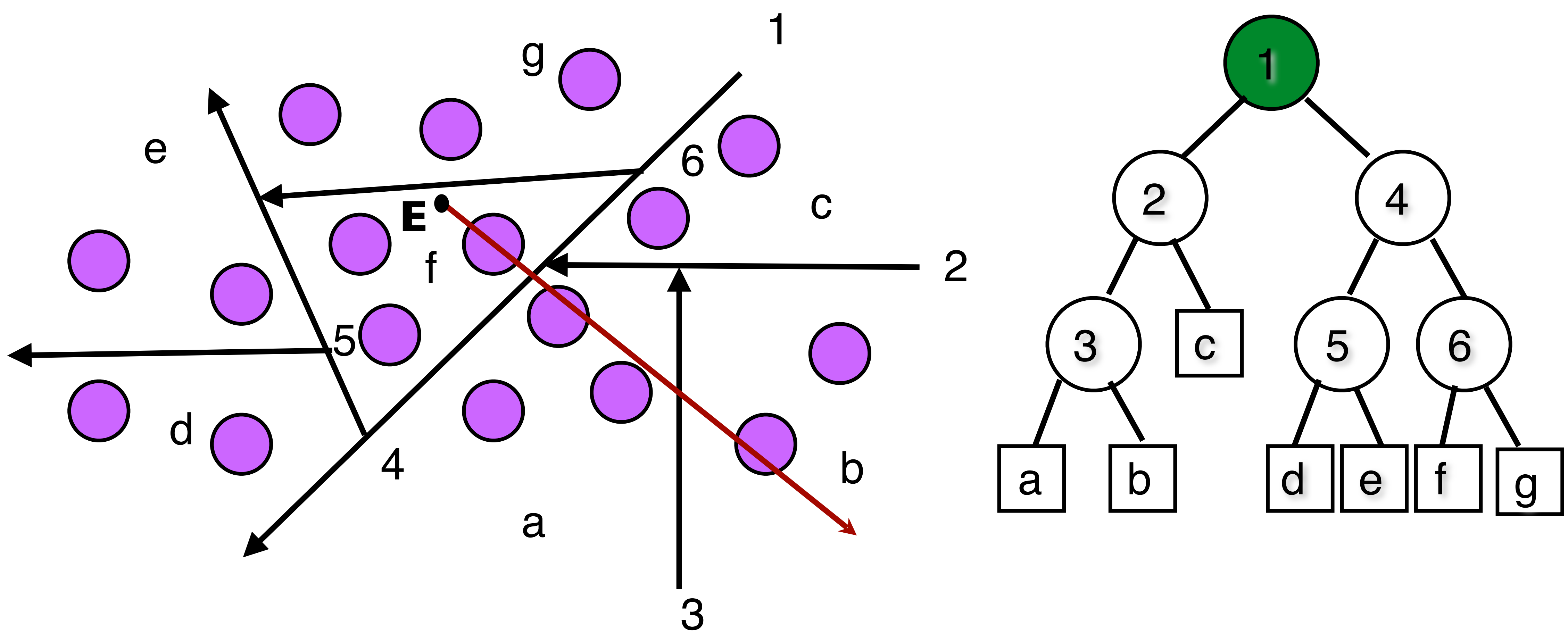
Traversal algorithm

```
visit(E, v, node): (E eye)
  if (node is leaf):
     intersect ray with objs in leaf
  else:
     if (E on left):
         visit(E, v, left)
         other = right;
     else:
         visit(E, v, right)
         other = left
     endif

     if (ray crosses boundary):
         E' = intersect(E, v, boundary)
         visit(E', v, other)
     endif

  endif
```

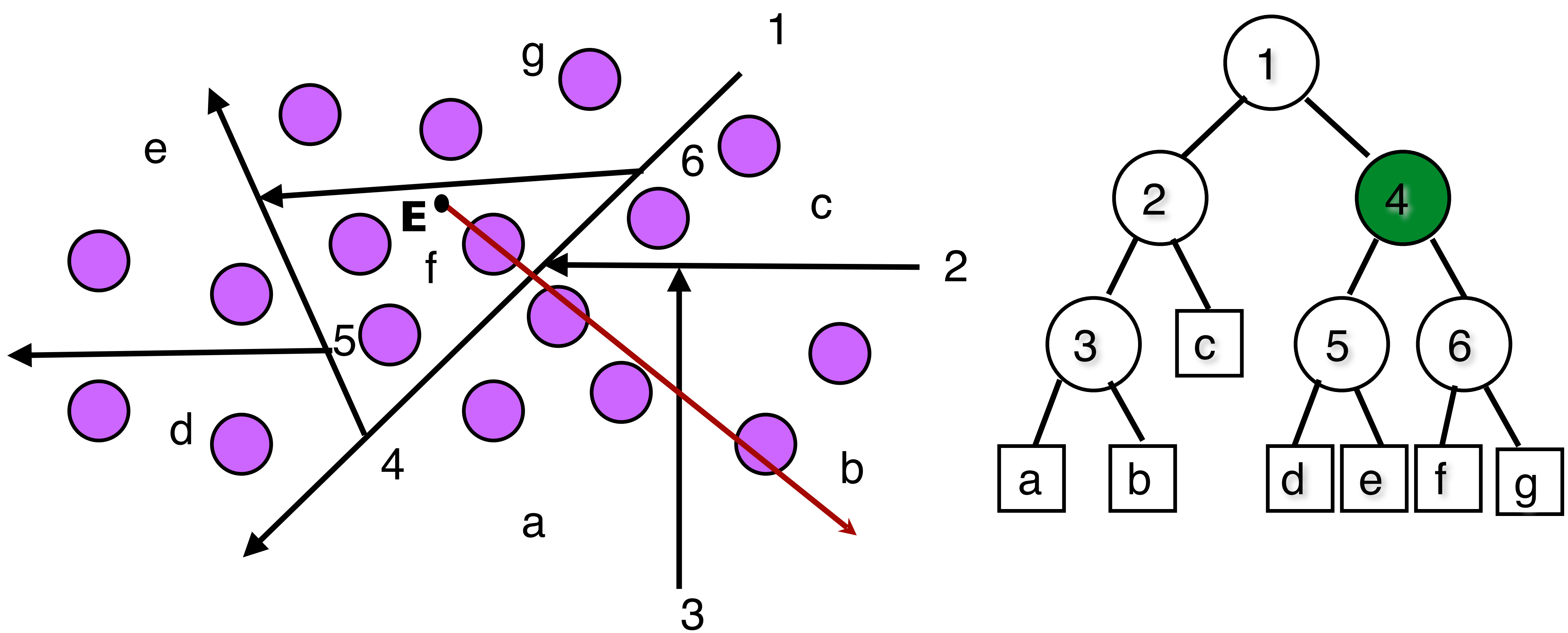
Traversing the tree



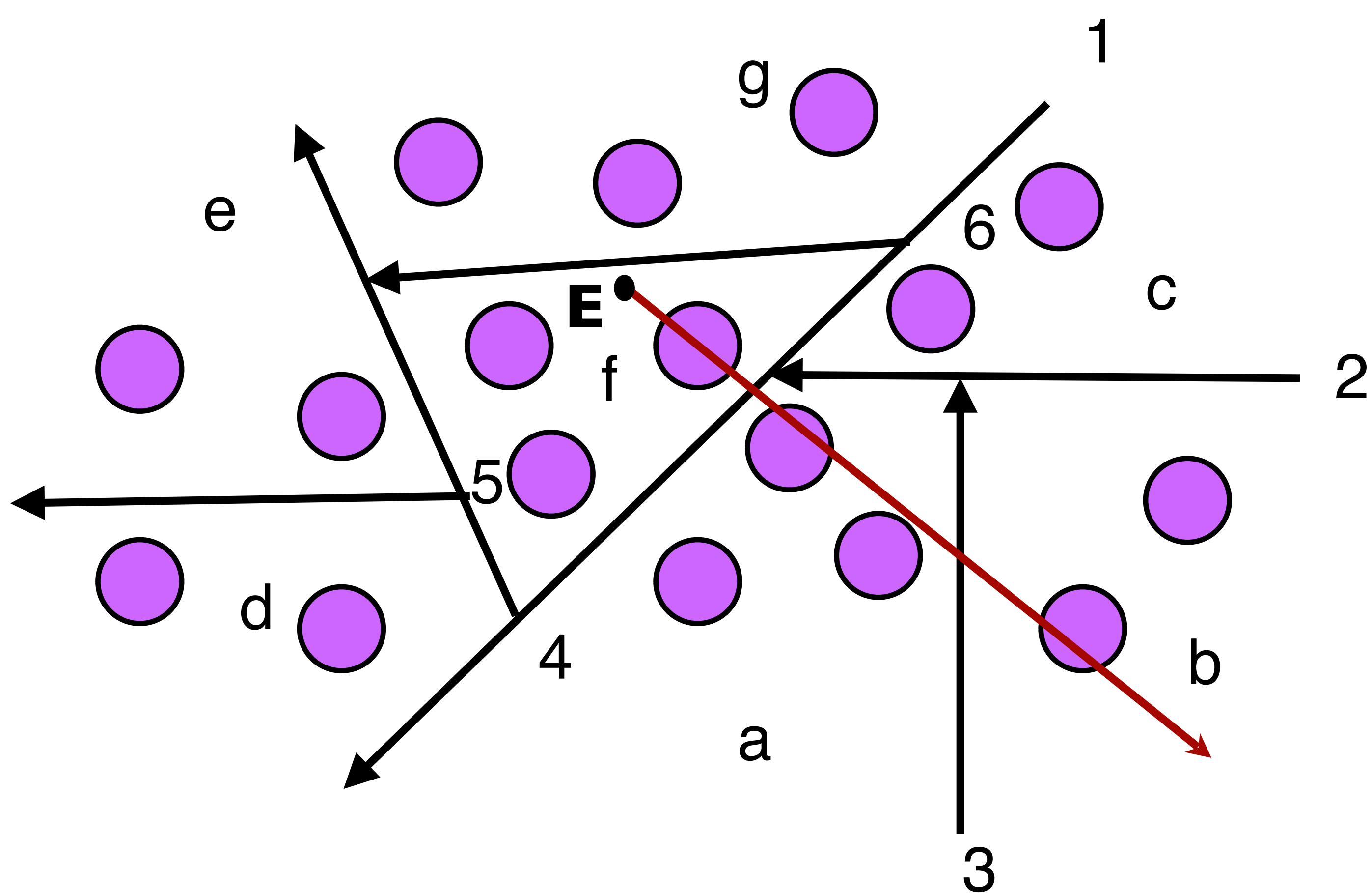
Traversing the tree

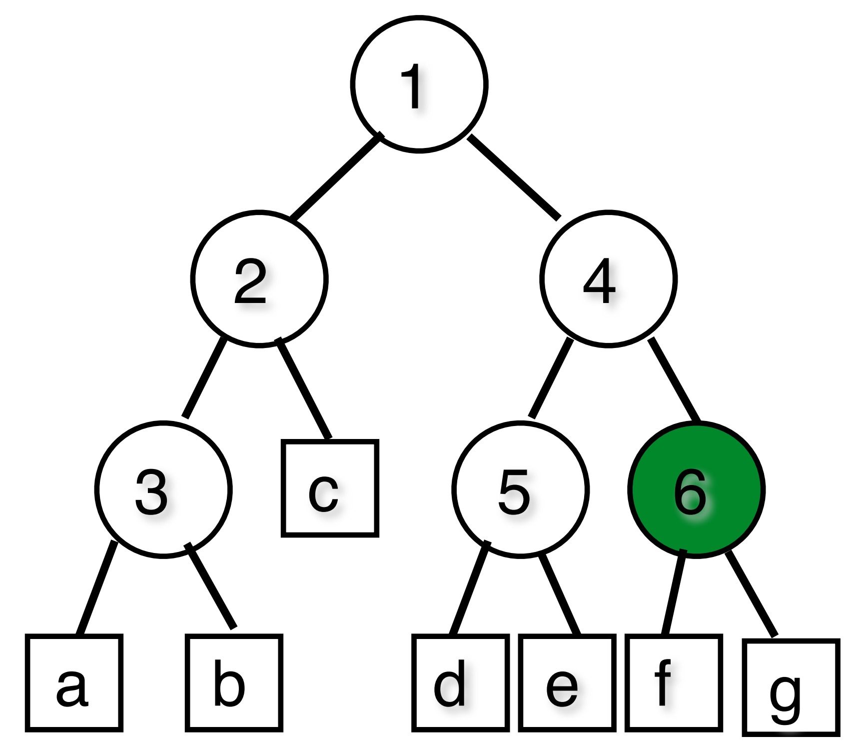


Traversing the tree

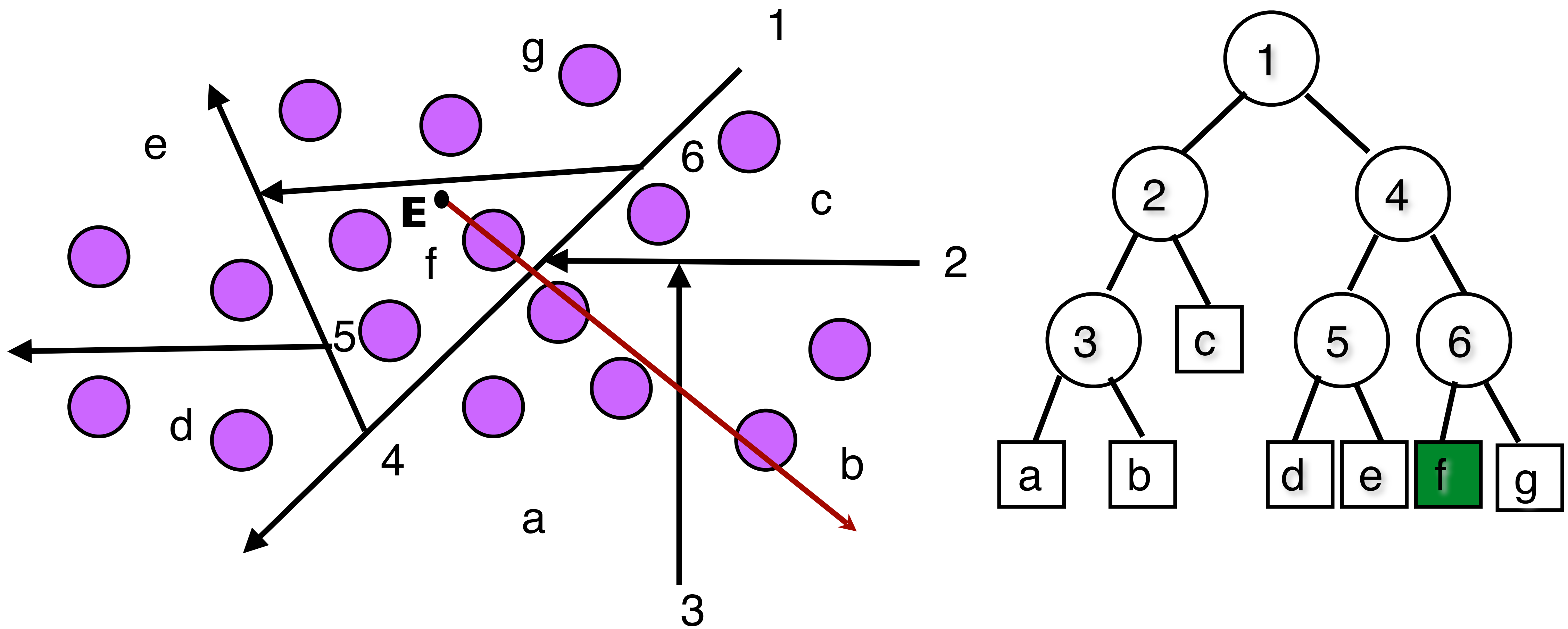


Traversing the tree

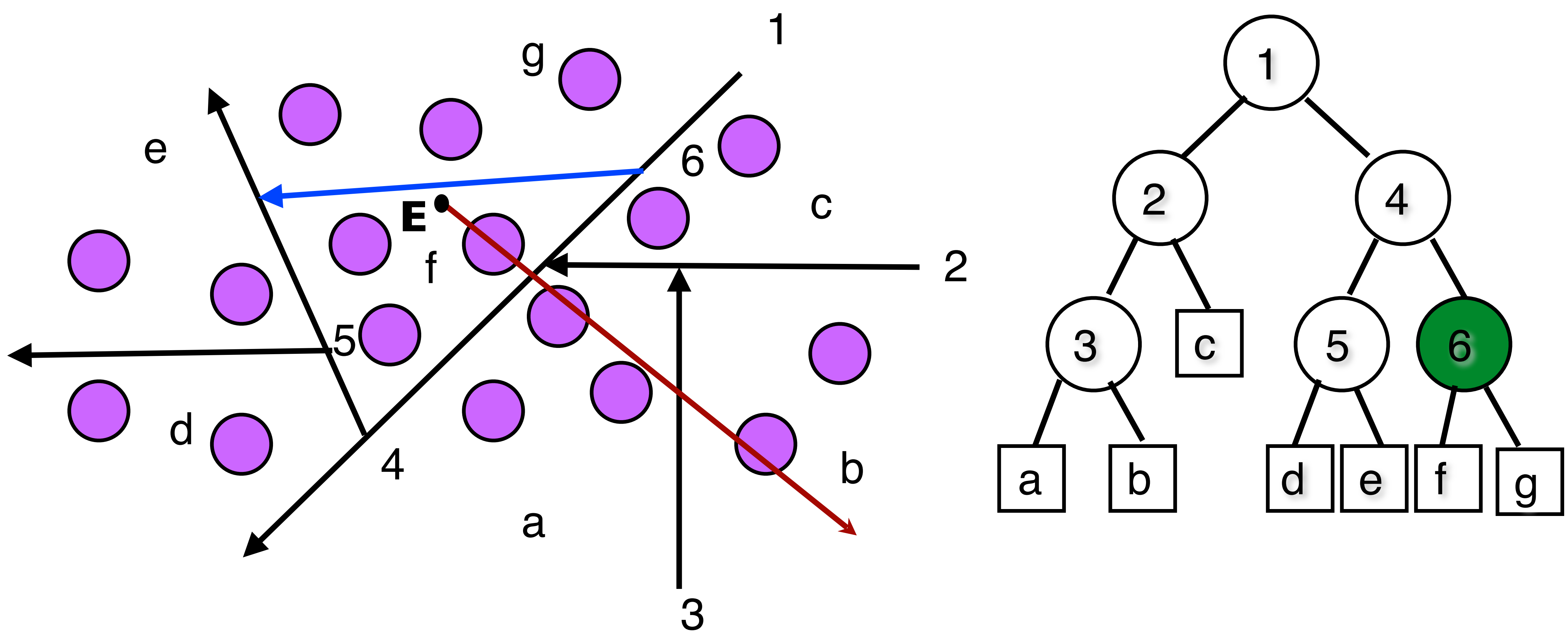




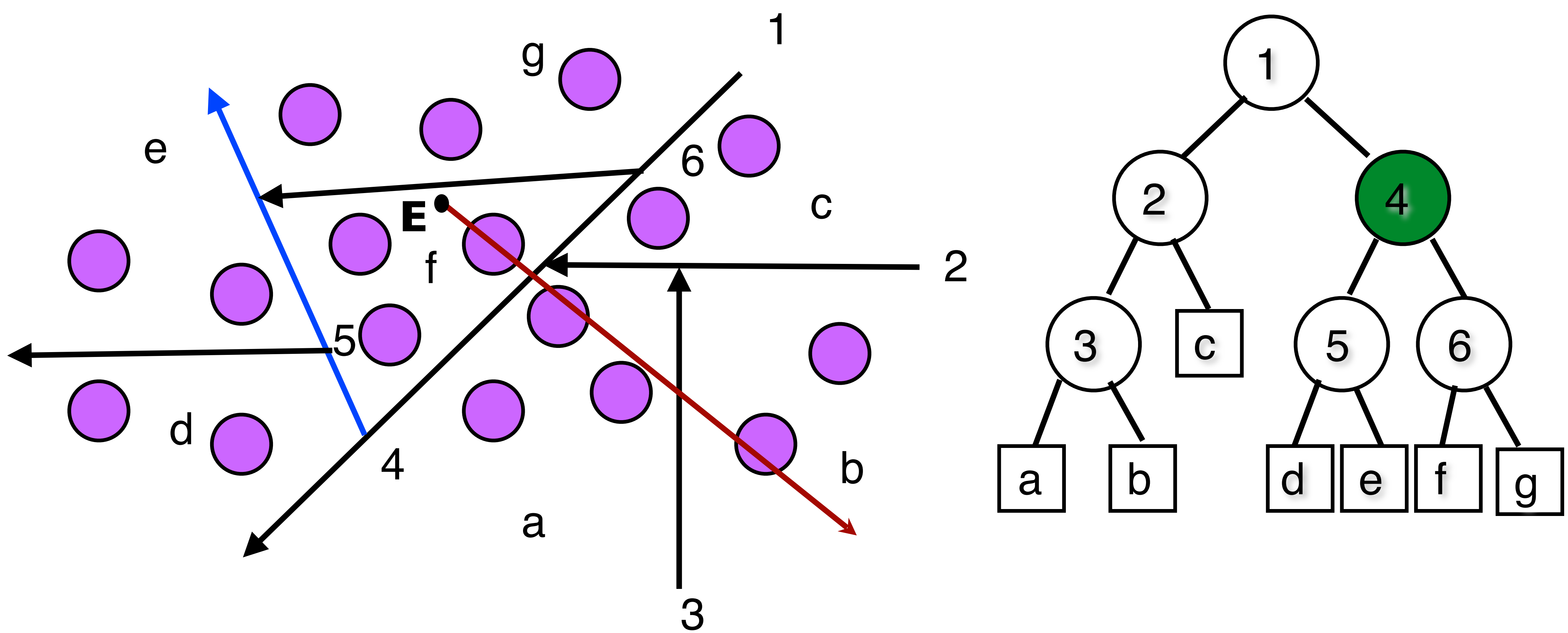
Traversing the tree



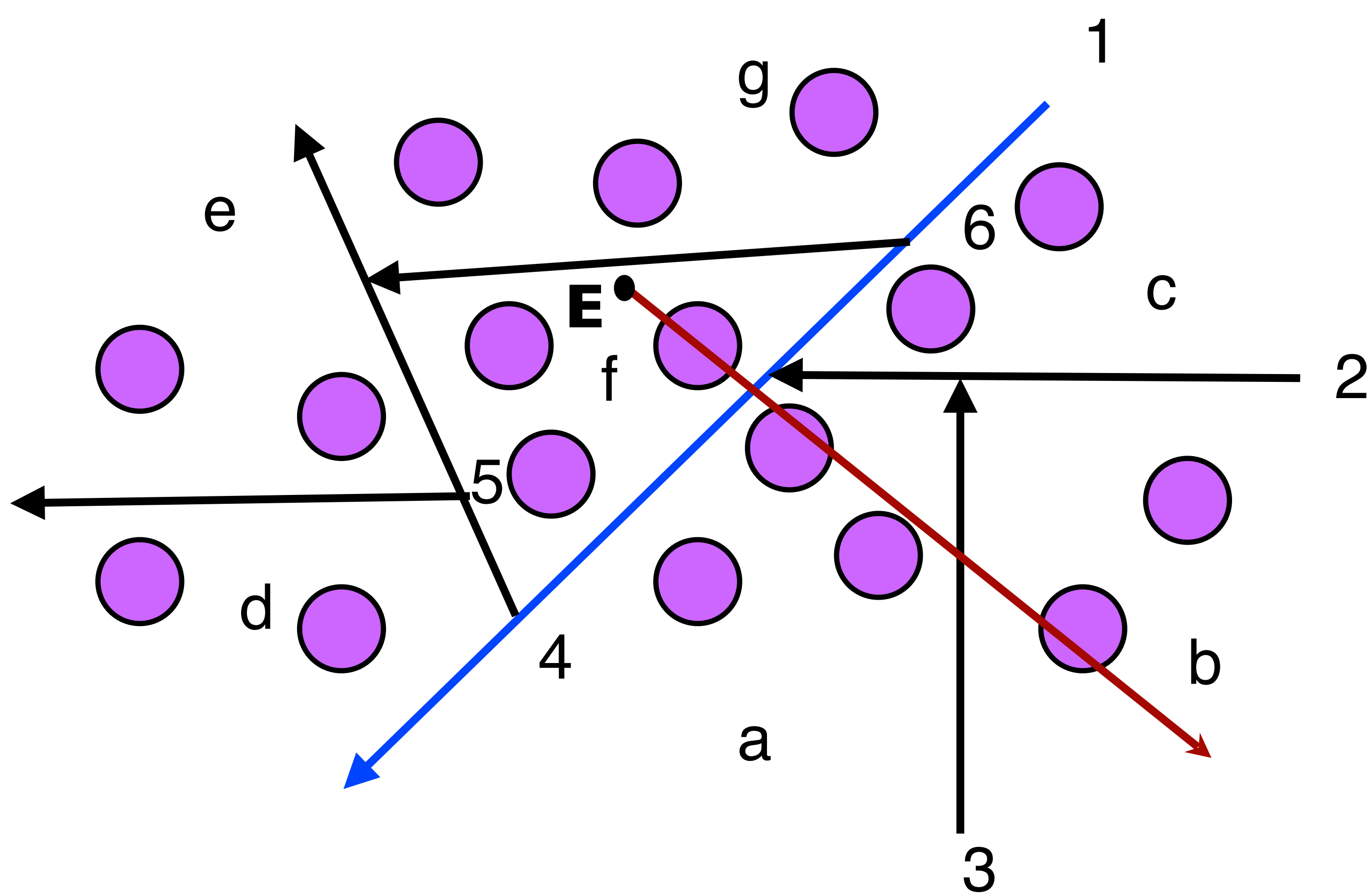
Traversing the tree

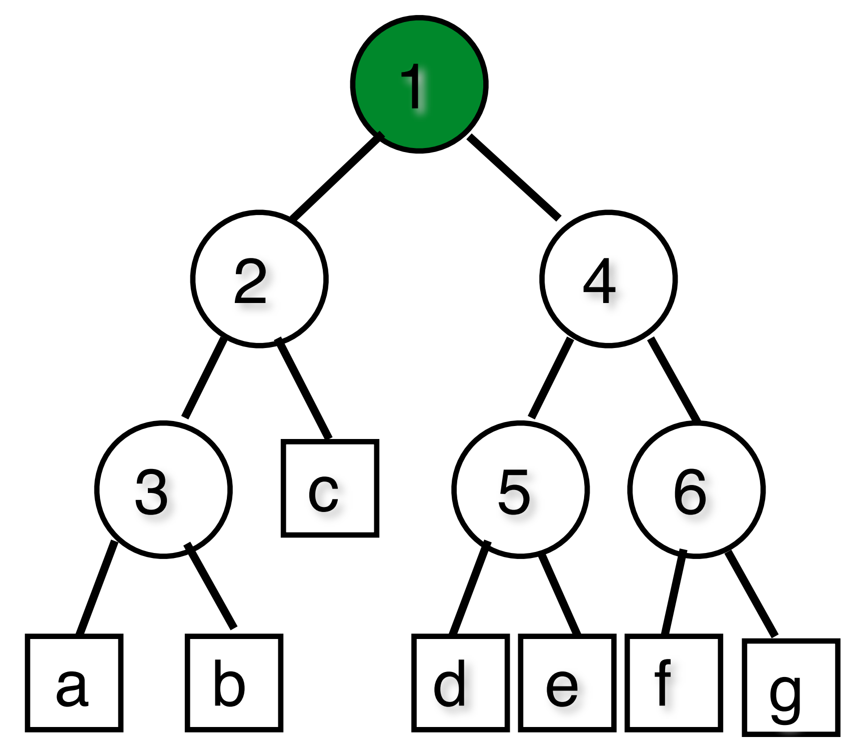


Traversing the tree

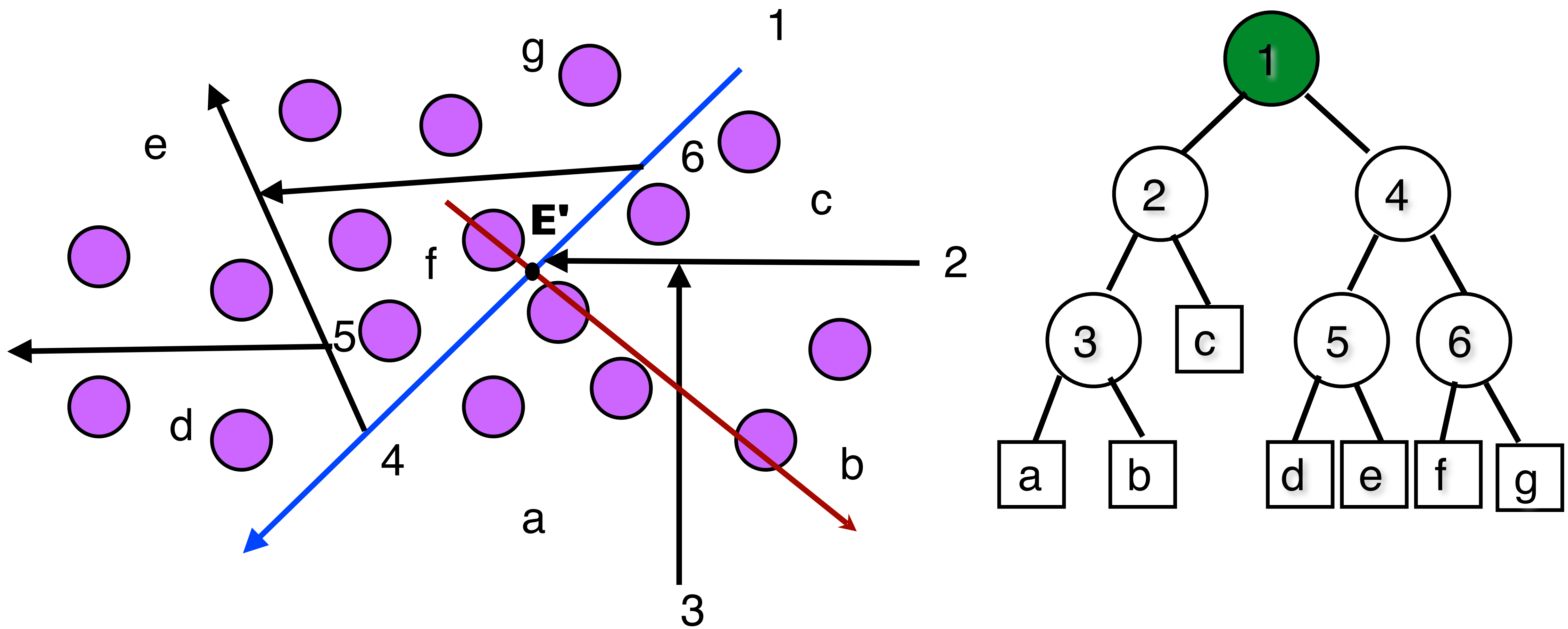


Traversing the tree

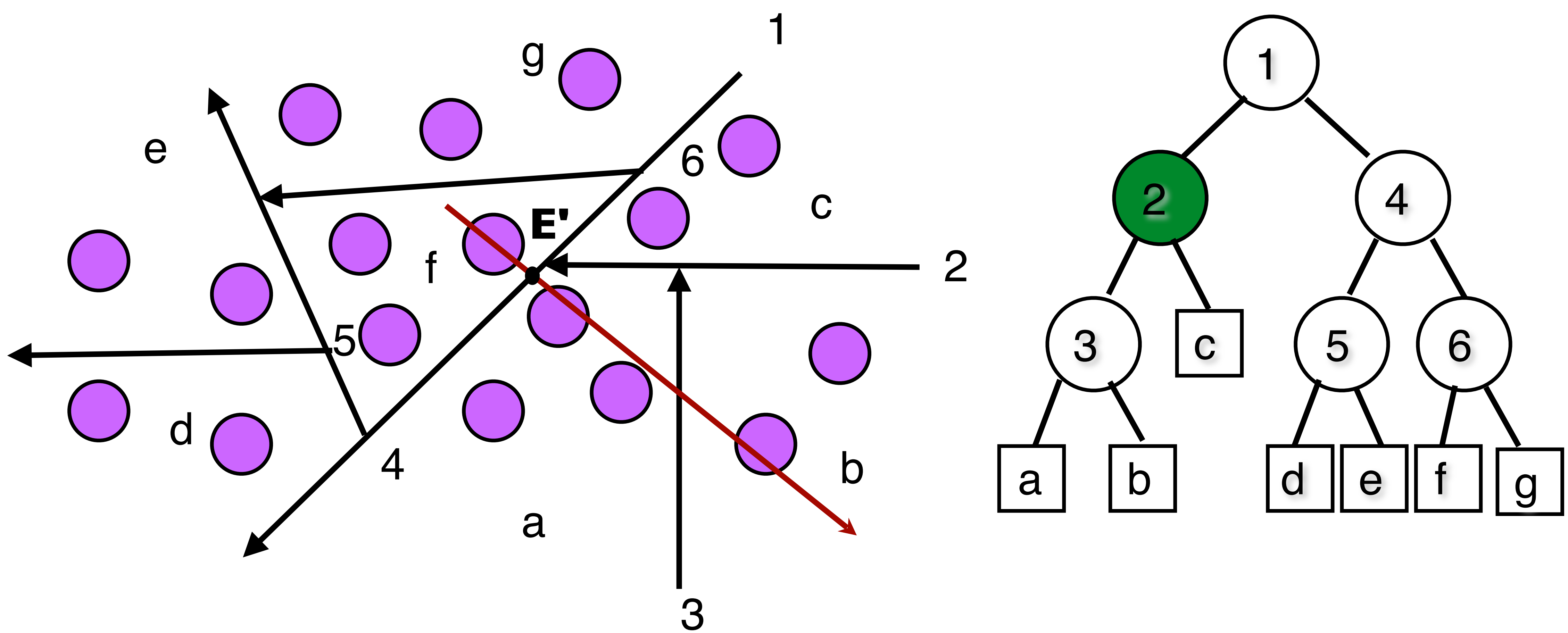




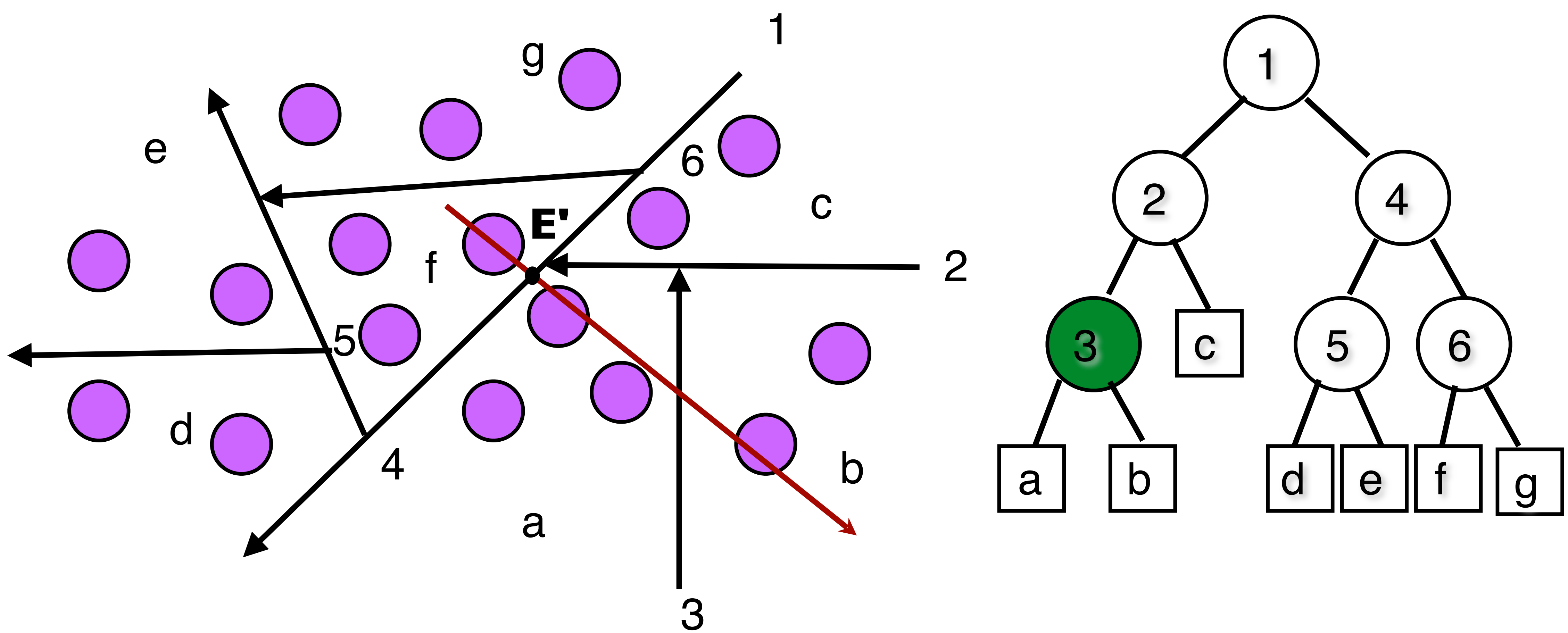
Traversing the tree



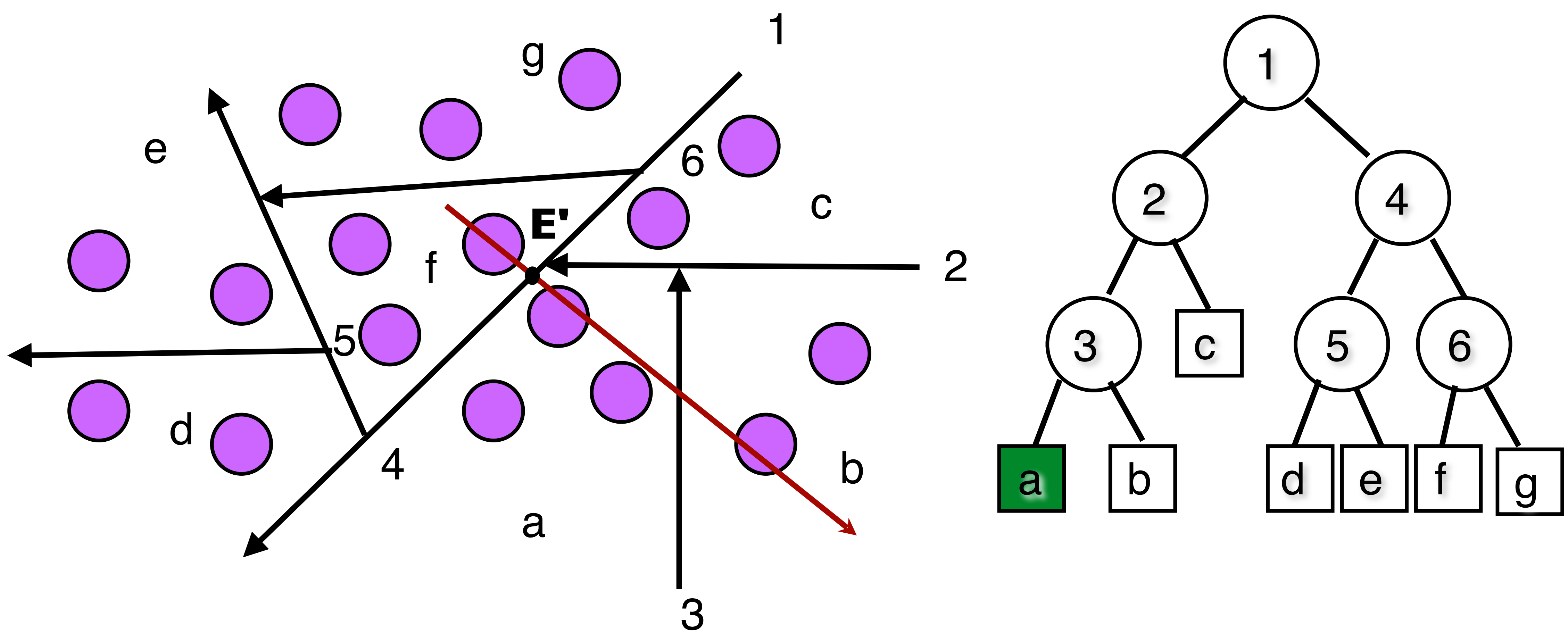
Traversing the tree



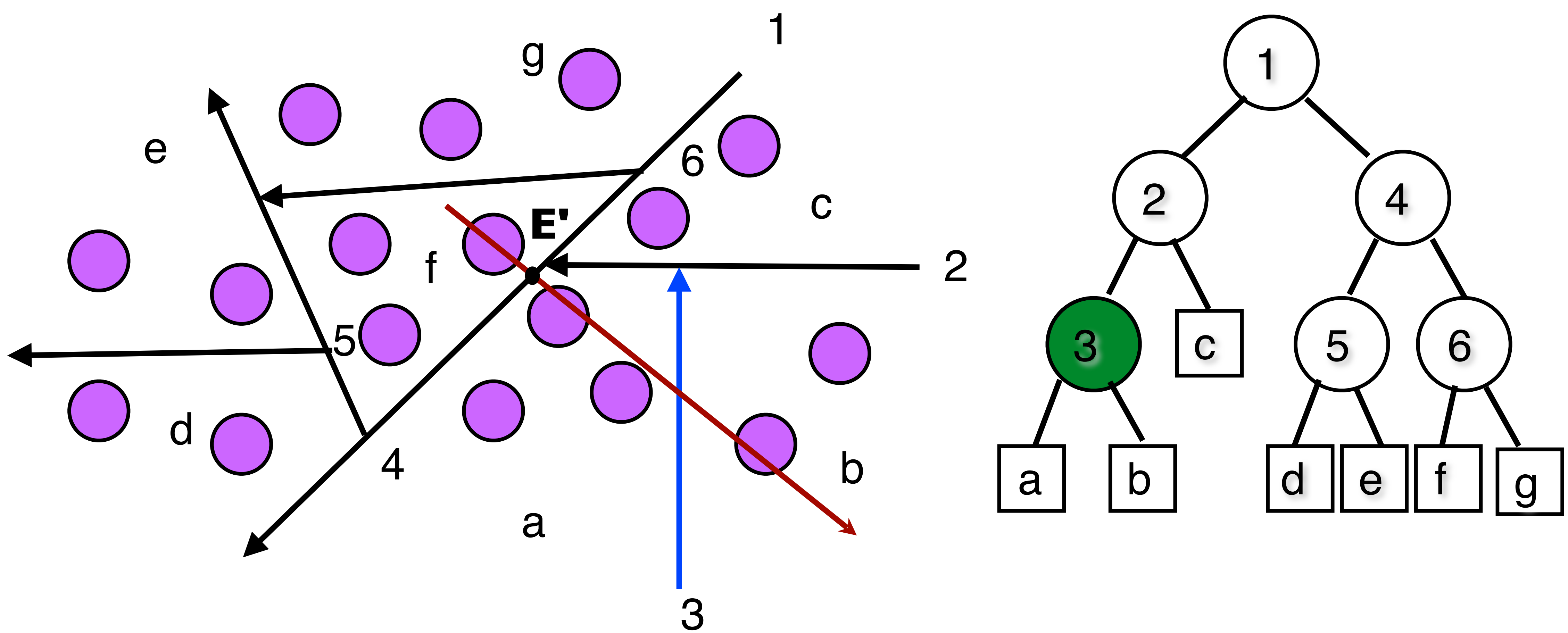
Traversing the tree



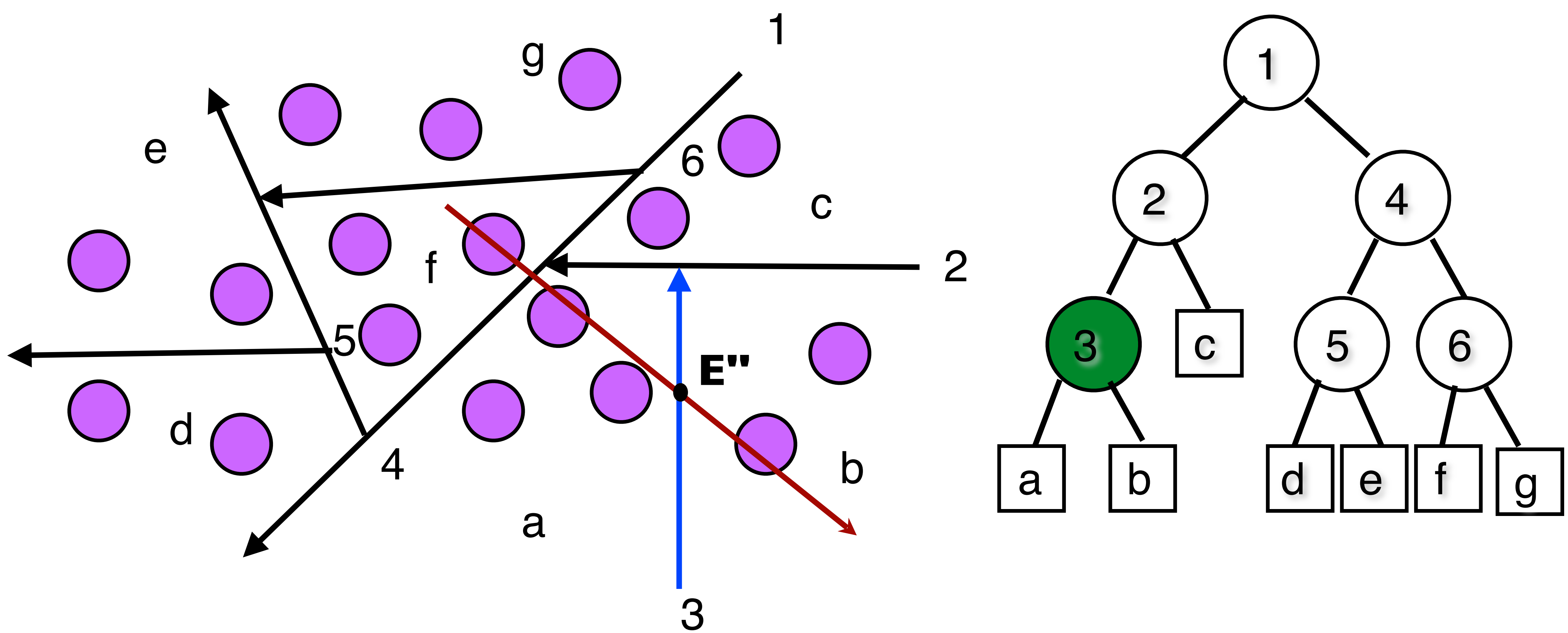
Traversing the tree



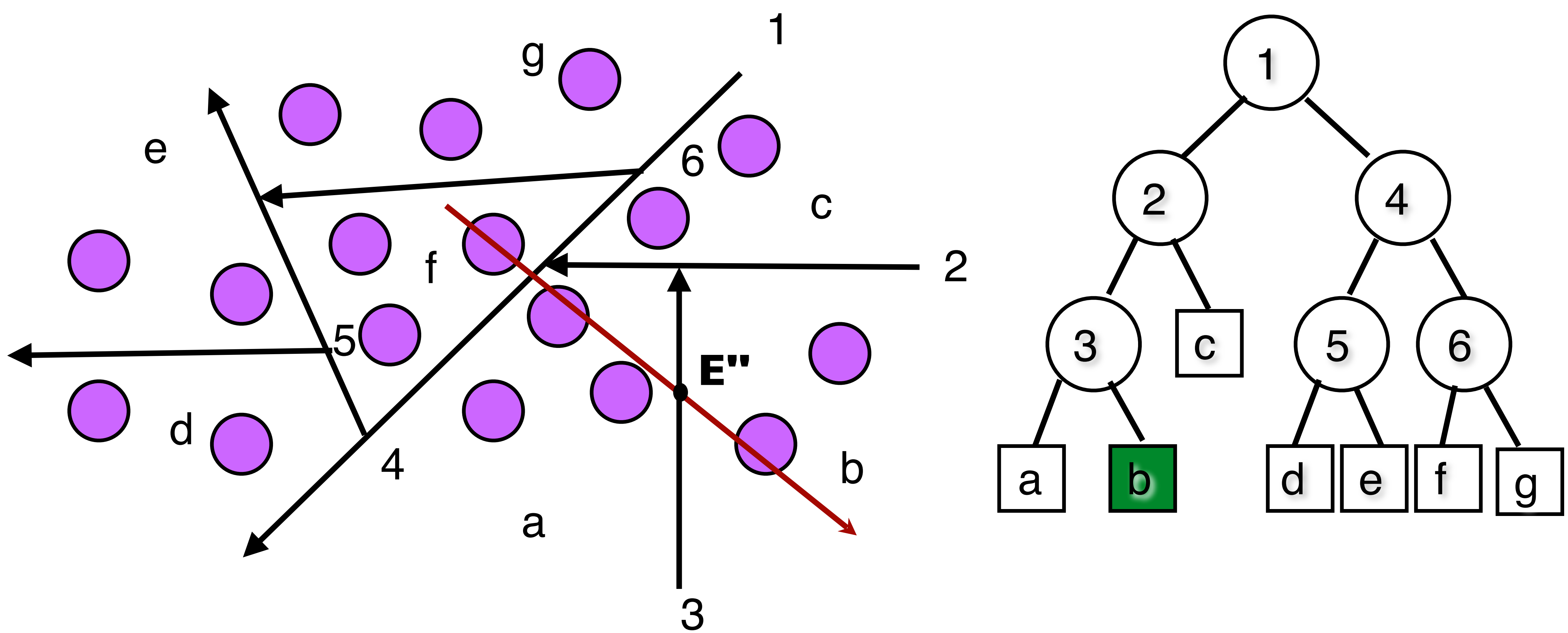
Traversing the tree



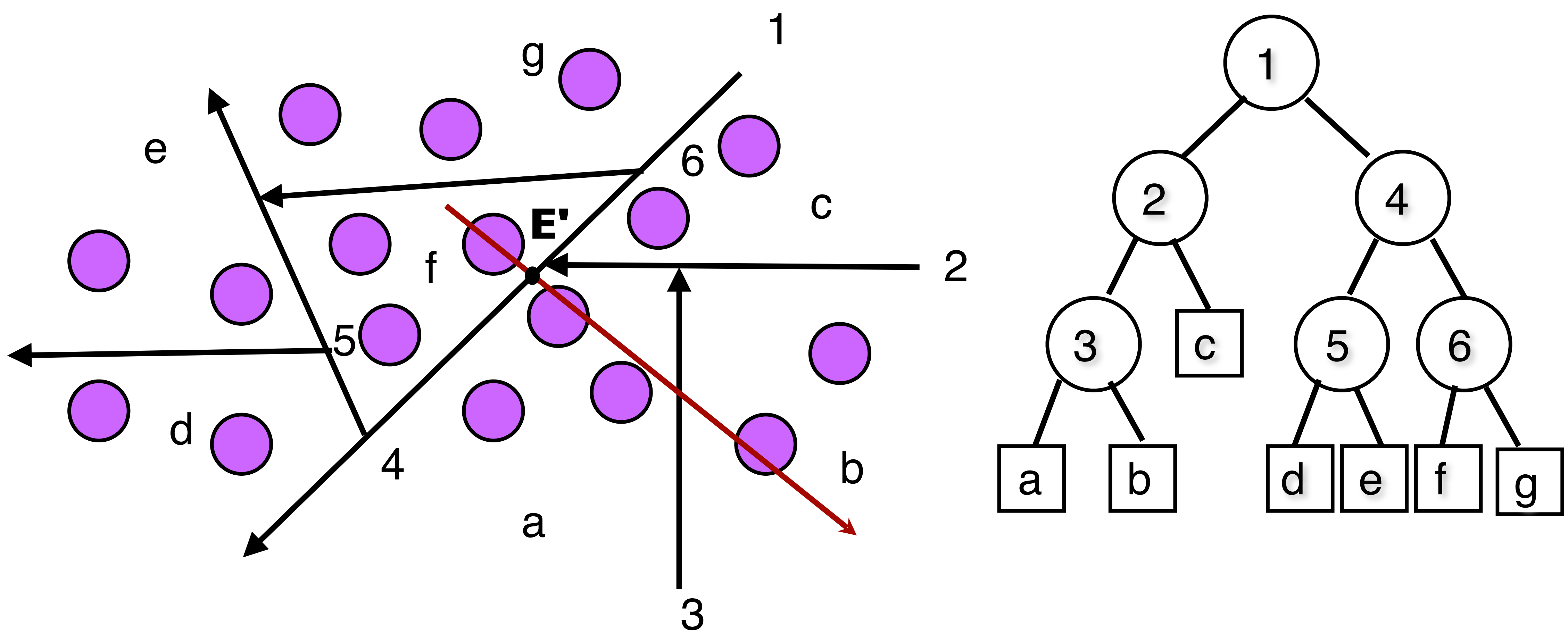
Traversing the tree



Traversing the tree



Traversing the tree



Exercise

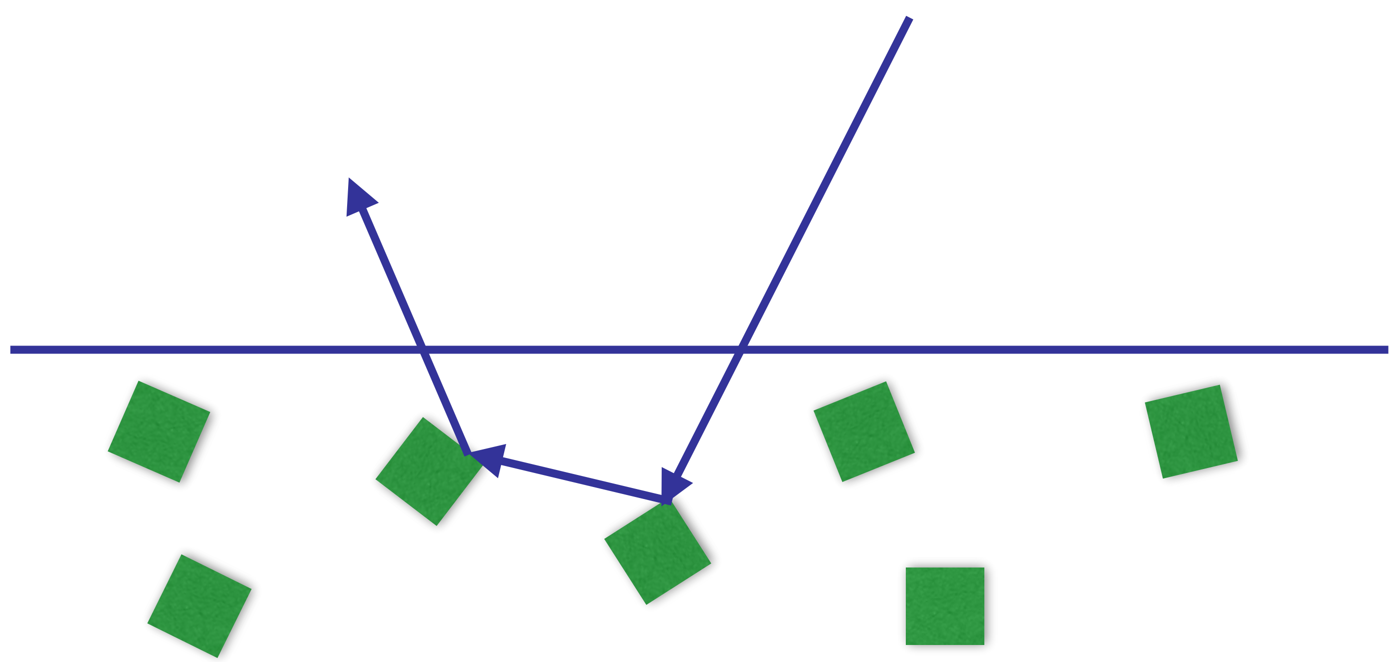
- How does milk look different to white paint?

Exercise

- How does milk look different to white paint?
- Both are opaque.
- Both are essentially pure white.
- Milk is an example of scattering

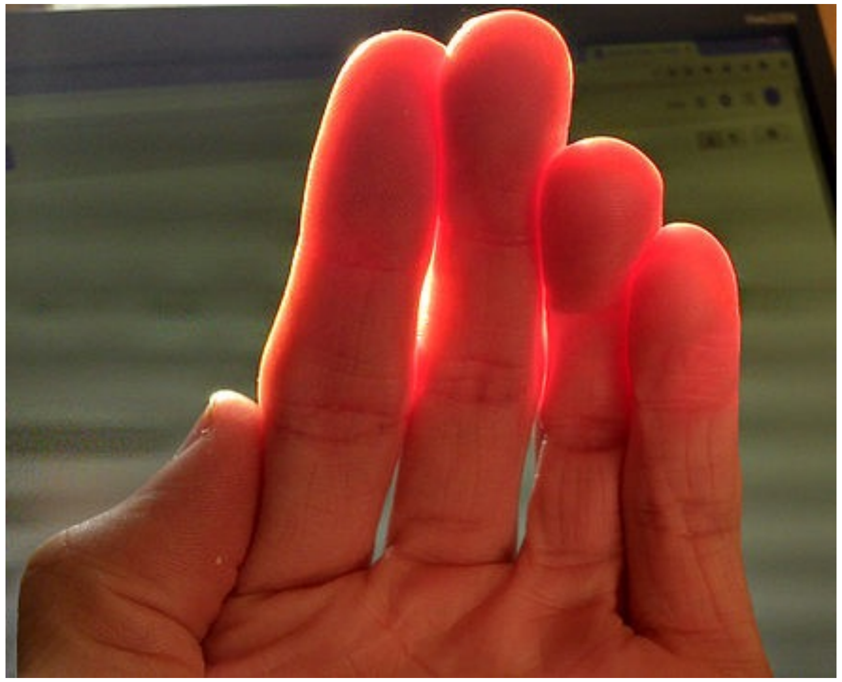
Scattering

- Scattering (or subsurface scattering) is when light refracts into an object that is non-uniform in its density and is reflected out at a different angle and position.



Scattering

- Milk is a substance that has this property.
- As is skin, leaves, and wax.
- Typically, they are hard to render.



Scattering

Scattering

- We don’t really have time to cover this in more depth in this course. Read this if you want to know more (NOT EXAMINABLE).
- http://graphics.ucsd.edu/~henrik/images/subsurf.html

Raytracing Can't Do

- Basic recursive raytracing cannot do:
  - Light bouncing off a shiny surface like a mirror and illuminating a diffuse surface
  - Light bouncing off one diffuse surface to illuminate others
  - Light transmitting then diffusing internally
- Also a problem for rough specular reflection
  - Fuzzy reflections in rough shiny objects

Realtime ray-tracing (RTX)

- Recent Nvidia innovation
- Realtime raytracing supported via specialised hardware
- Programs have to be written to use it, but it fits in quite well with existing graphics pipelines

Realtime ray-tracing (RTX)

- Works by arranging objects in a bounding volume hierarchy (BVH)
- Specialised hardware offers fast traversal of these hierarchies to find ray intersections.

Bounding Volume Hierarchy (BVH)

- Scene is divided into small volumes.
- Child volumes are contained entirely within their parents.
- Sibling volumes may overlap

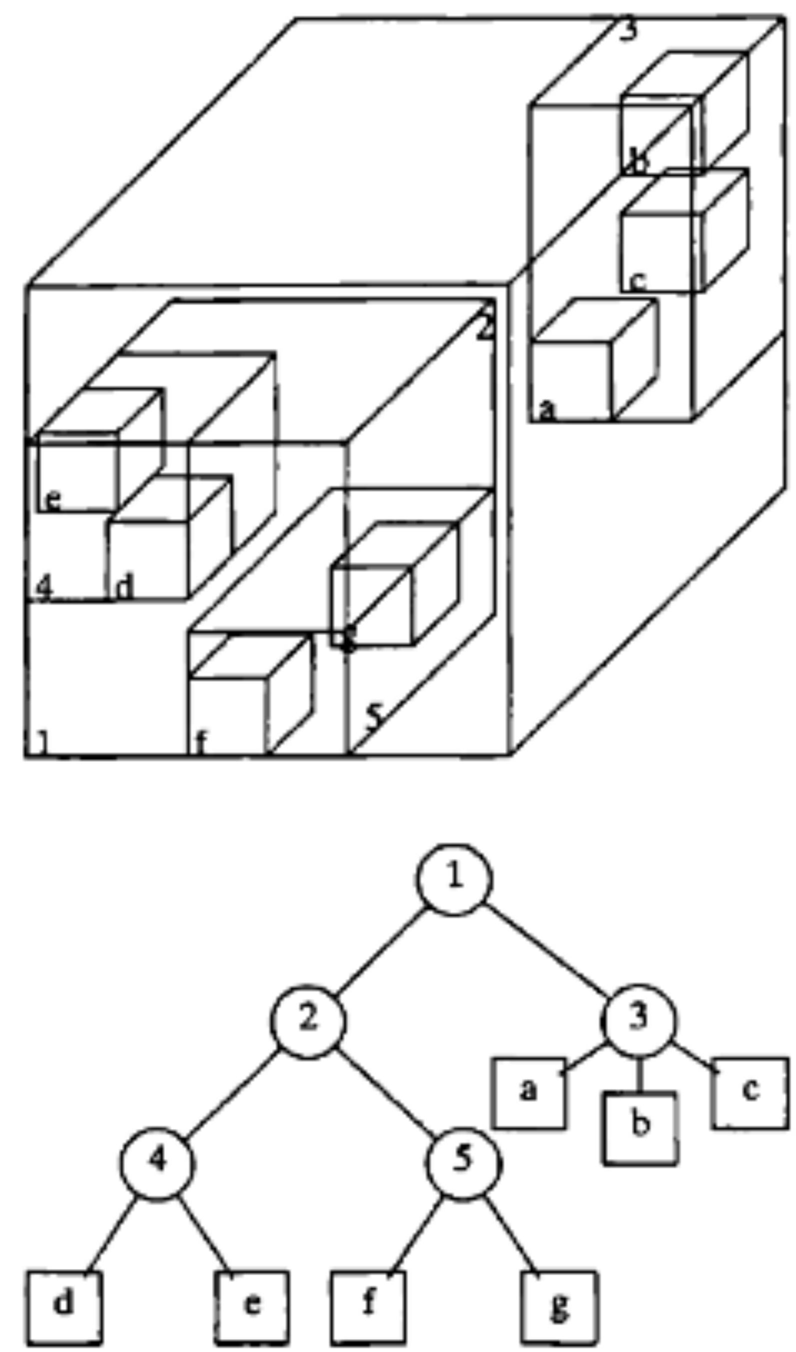


Figure 2.2
Letters a through g represent the objects.
Numbers 1 through 5 represent interior nodes.

Bounding Volume Hierarchy (BVH)

- Parents can have an arbitrary number of children
- The best way to divide up the scene depends on a lot of factors

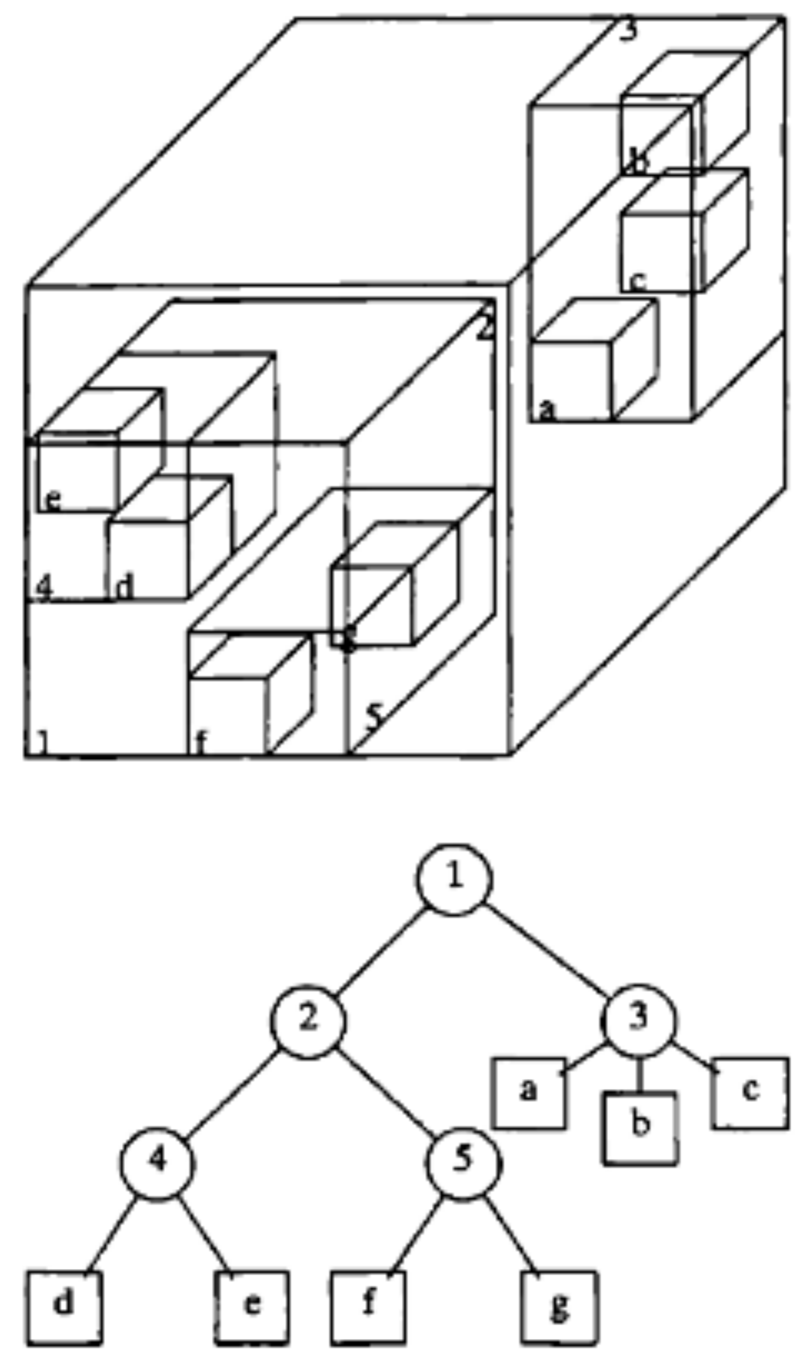


Figure 2.2
Letters a through g represent the objects.
Numbers 1 through 5 represent interior nodes.

Top down construction

- Divide the set of primitives in the scene into two (or more) subsets then recursively subdivide those until all subsets contain only one primitive.
- Easy to implement.
- Usually faster than alternatives.
- Doesn’t always produce the best possible tree.

Bottom up construction

- Treat all primitives in the scene as a set of leaves. Pick two (or more) of them and group them into a node. Repeat till there is only one node in the set.

- Slower than top-down in most cases.

- More difficult to implement.

- Tends to produce better trees.

RTX Examples

- https://www.youtube.com/watch?v=WoQr0k2IA9A
- https://www.youtube.com/watch?v=Ms7d-3Dprio
- https://www.youtube.com/watch?v=1IliQZw_p_E

Volumetric ray tracing

- We can also apply ray tracing to volumetric objects like smoke or fog or fire.
- Such objects are transparent, but have different intensity and transparency throughout the volume.

Volumetric Ray Tracing

- We represent the volume as two functions:

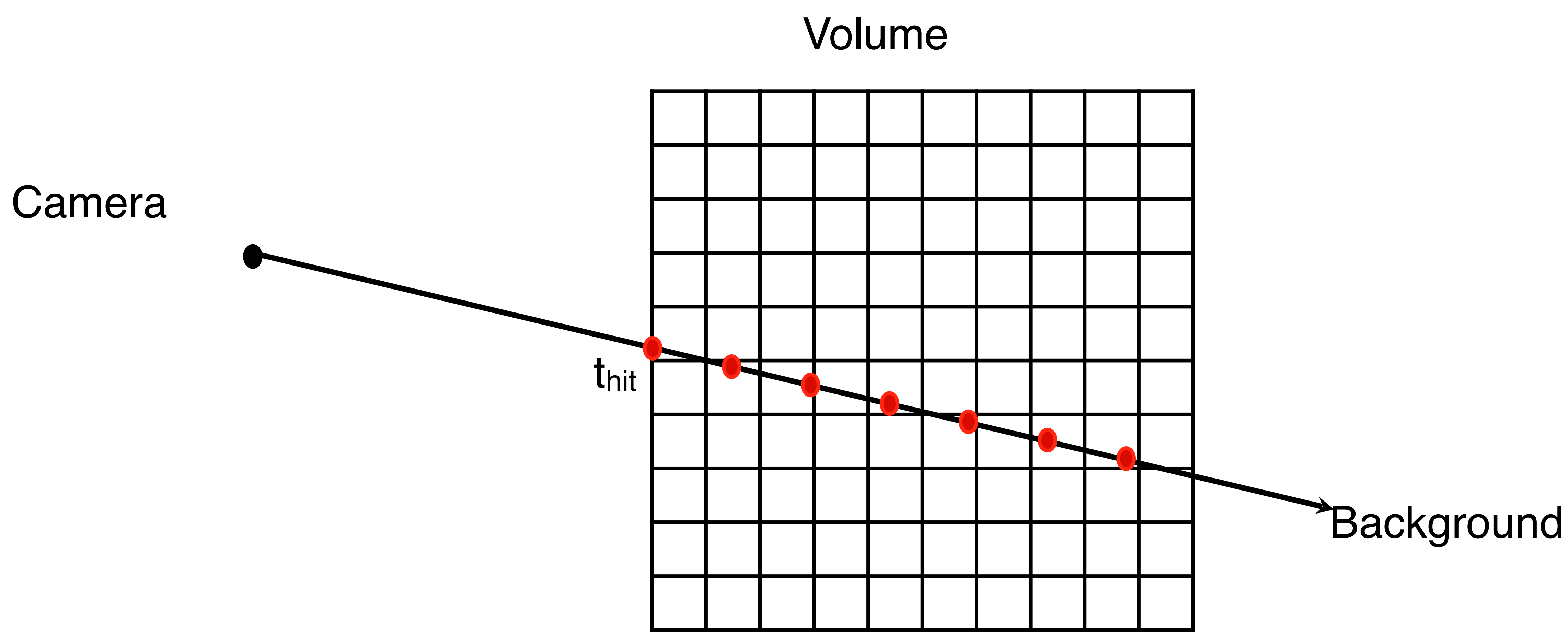
$$
\begin{aligned}
C(P) &= \text{colour at point } P \\
\alpha(P) &= \text{transparency at point } P
\end{aligned}
$$

- Typically these are represented as values in a 3D array. Interpolation is used to find values at intermediate points.

- These functions may in turn be computed based on density, lighting or other physical properties.

Sampling

- We cast a ray from the camera through the volume and take samples at fixed intervals along the ray.



Sampling

- We end up with (N+1) samples:

$$
\begin{aligned}
P_i &= R(t_{hit} + i\Delta t) \\
C_i &= C(P_i) \\
\alpha_i &= \alpha(P_i) \\
C_N &= (r, g, b)_{background} \\
\alpha_N &= 1
\end{aligned}
$$

Alpha compositing

- We now combine these values into a single colour by applying the alpha-blending equation.

$$C_N^N = C_N$$

$$\underbrace{C_N^i}_{\text{Total colour at i}} = \alpha_i \underbrace{C_i}_{\text{Local colour at i}} + (1 - \alpha_i) \underbrace{C_N^{i+1}}_{\text{Total colour at i+1}}$$

Exercise

- Suppose we have a background color of (0,1,0) and a volume with the uniform color of (1,0.5,0.5). A ray cast through that volume takes two samples. The first has an alpha value of 0.2 and the second 0.1. What is the colour of the resulting pixel?

Alpha compositing

- We can write a closed formula for the colour from a to b as:

$$C_b^a = \sum_{i=a}^{b} \alpha_i C_i \prod_{j=a}^{i-1} (1 - \alpha_j)$$

- We can compute this function from front to back, stopping early if the transparency term gets small enough that nothing more can be seen.

In OpenGL

- Volumetric ray tracing (aka ray casting) does not require a full ray tracing engine.
- It can be implemented in OpenGL as a fragment shader applied to a cube with a 3D texture.
- https://www.shadertoy.com/view/XslGRr

Examples

- https://www.shadertoy.com/view/Ms2SD1
- https://www.shadertoy.com/view/4sS3zG
- See http://shadertoy.com/ for more examples.

Sources

- http://en.wikipedia.org/wiki/Volume_ray_casting
- http://graphics.ethz.ch/teaching/former/scivis_07/Notes/Slides/03-raycasting.pdf
- http://http.developer.nvidia.com/GPUGems/gpugems_ch39.html