Het kleurtjesalgoritme helpt de planner om een besluit te maken over een aanvraag.

Aan de front end ziet de planner elke nieuwe aanvraag die werd gedaan, omringd door een kleurtje. Wanneer het een rood kleurtje heeft, is er geen planning mogelijk wanneer die persoon op verlof wordt gestuurd. Deze aanvraag mag dus nooit worden goedgekeurd.

Als de aanvraag een groen kleurtje heeft, dan mag deze aanvraag altijd worden goedgekeurd. Ze zorgt nooit voor problemen.

Als de aanvraag oranje is, is het ingewikkelder. Stel dat er twee personen zijn die een bepaalde shift kunnen doen en er één persoon nodig is om de shift te doen. Eén van de twee mag dan worden op verlof gestuurd. Ze mogen echter nooit allebei tegelijk worden op verlof gestuurd, want dan kan er niemand de shift invullen. Ze vormen een groep waarvan één persoon niet op verlof mag gaan.

De uitdaging van dit algoritme is nu net om al deze groepen te vinden.

Om het algoritme makkelijk te kunnen uitleggen, worden de aanvragen makkelijker voorgesteld doormiddel van cijfertjes. Elke groep krijgt een cijfertje. Alle elementen van dezelfde groep, krijgen hetzelfde cijfertje.

Nu kunnen we het probleem voorstellen in een array waarin elk element één of meerdere cijfertjes bevat. Bovenaan zien we de array met de te vinden oplossing. Die eronder is wat we al weten, momenteel dus niets.

We bezitten slechts over één soort test: kunnen we een bepaalde groep op verlof sturen en kan er dan nog een planning worden gemaakt? Bij de getalletjes komt dit neer op: “Bezit een te testen groep elk element van een bepaald cijfertje.”

Nu zullen we op zoek gaan naar de groepen.

Daarvoor testen we eerst de volledige groep, als deze geen groepen bevat, dan mogen ze allemaal worden goedgekeurd, er is geen probleem.

In ons geval is er wel een probleem, dus zullen we verder testen.

In een tweede stap testen we nu eens elk element apart. Zorgt een elementje testen op zichzelf voor een probleem, dan mag het nooit worden goedgekeurd.

Deze elementen zullen als rood worden bestempeld en worden uit de rest van de testen uitgelaten.

Nu testen we nog eens de volledige groep, misschien zitten er geen oranje in. In dat geval mogen de elementjes ook als groen worden teruggegeven. In ons geval crasht de groep nog steeds.

We weten dus nu zeker dat er nog een groep is die groter dan 1 is.

Nu gaan we op zoek naar de resterende groepen

Daarom testen we eerst de eerste twee elementen. Deze zijn oké, dus kijken we naar de eerste drie. Deze crashen wel.

Het laatste element van deze drie zorgt voor de crash, dus is zeker lid van een groepje. Nu laten we het eerste en tweede element eens uit de groep en testen we deze groepen. Als het eerste er wordt uitgelaten, crasht de groep niet meer. Het eerste element is dus ook lid van de groep. Als het tweede er wordt uitgelaten, crasht de groep wel nog, het was dus geen lid van die groep.

Nu willen we de groep nog groter maken om nieuwe groepjes te vinden. Wanneer we echter gewoon grootte 4 zouden nemen, crasht de groep nog steeds, door de aanwezigheid van groep 4. Daarom schakelen we het eerste element van groep 4 uit.

Nu maken we de groep weer groter tot hij crasht. Zo wordt groep 6 analoog aan groep 4 gevonden.

Nu zijn we aan het einde van de array en hebben we groep 5 niet gevonden. Dit komt doordat we telkens de eerste elementen uit een groepje weglieten. De oplossing houdt in dat we nu ook eens de volledige groep testen, maar met telkens een andere permutatie van elementjes die uit de groep worden gelaten.

Zo werd nu BF uitgelaten, wordt dan BG, dan DF en dan DG getest, we kunnen dit voorstellen onder een binaire manier als 00 01 10 11.

Wanneer DG uitgelaten wordt, kan groep 5 worden gevonden.

In principe wordt nu nog eens getest met alle permutaties die groep 5 inhouden.

Dan zijn eindelijk alle groepen gevonden.

De performantie werd nog verbeterd door in de laatste stap met permutaties te kijken of van de groep al een supergroep getest is. Want met meerdere groepen kunnen verschillende permutaties op dezelfde groep slaan.

Ook wordt de groep in het begin gesplitst in meerdere subgroepjes. Deze subgroepjes kunnen sowieso nooit invloed op elkaar hebben omdat er tussen hen meer dagen liggen dan er maximaal aantal dagen recup kunnen zijn.

Recup houdt in: als iemand maandag een shift doet met 3 dagen recup, dan mag hij de volgende 3 dagen; dinsdag, woensdag en donderdag; niet werken.

Dit zorgt ervoor dat de stack van de backtracker die zich in de test bevindt opmerkelijk kleiner wordt.